TutorialsPoint 其它教程

wizardforcel

Published with GitBook



目錄

TutorialsPoint 其它教程

W3School 设计模式教程

来源：设计模式教程

整理：飞龙

设计模式简介

设计模式（Design pattern）代表了最佳的实践，通常被有经验的面向对象的软件开发人员所采用。设计模式是软件开发人员在软件开发过程中面临的一般问题的解决方案。这些解决方案是众多软件开发人员经过相当长的一段时间的试验和错误总结出来的。

设计模式是一套被反复使用的、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了重用代码、让代码更容易被他人理解、保证代码可靠性。 毫无疑问，设计模式于己于他人于系统都是多赢的，设计模式使代码编制真正工程化，设计模式是软件工程的基石，如同大厦的一块块砖石一样。项目中合理地运用设计模式可以完美地解决很多问题，每种模式在现实中都有相应的原理来与之对应，每种模式都描述了一个在我们周围不断重复发生的问题，以及该问题的核心解决方案，这也是设计模式能被广泛应用的原因。

什么是 **GOF**（四人帮，全拼 **Gang of Four**）？

在 1994 年，由 Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 四人合著出版了一本名为 **Design Patterns - Elements of Reusable Object-Oriented Software**（中文译名：设计模式 **-** 可复用的面向对象软件元素） 的书，该书首次提到了软件开发中设计模式的概念。

四位作者合称 **GOF**（四人帮，全拼 **Gang of Four**）。他们所提出的设计模式主要是基于以下的面向对象设计原则。

- 对接口编程而不是对实现编程。
- 优先使用对象组合而不是继承。

设计模式的使用

设计模式在软件开发中的两个主要用途。

开发人员的共同平台

设计模式提供了一个标准的术语系统，且具体到特定的情景。例如，单例设计模式意味着使用单个对象，这样所有熟悉单例设计模式的开发人员都能使用单个对象，并且可以通过这种方式告诉对方，程序使用的是单例模式。

最佳的实践

设计模式已经经历了很长一段时间的发展，它们提供了软件开发过程中面临的一般问题的最佳解决方案。学习这些模式有助于经验不足的开发人员通过一种简单快捷的方式来学习软件设计。

设计模式的类型

根据设计模式的参考书 **Design Patterns - Elements of Reusable Object-Oriented Software**（中文译名：设计模式 **-** 可复用的面向对象软件元素） 中所提到的，总共有 23 种设计模式。这些模式可以分为三大类：创建型模式（Creational Patterns）、结构型模式（Structural Patterns）、行为型模式（Behavioral Patterns）。当然，我们还会讨论另一类设计模式：J2EE 设计模式。

模式 & 描述	包括
创建型模式 这些设计模式提供了一种在创建对象的同时隐藏创建逻辑的方式，而不是使用新的运算符直接实例化对象。这使得程序在判断针对某个给定实例需要创建哪些对象时更加灵活。	工厂模式（Factory Pattern） 抽象工厂模式（Abstract Factory Pattern） 单例模式（Singleton Pattern） 建造者模式（Builder Pattern） 原型模式（Prototype Pattern）
结构型模式 这些设计模式关注类和对象的组合。继承的概念被用来组合接口和定义组合对象获得新功能的方式。	适配器模式（Adapter Pattern） 桥接模式（Bridge Pattern） 过滤器模式（Filter、Criteria Pattern） 组合模式（Composite Pattern） 装饰器模式（Decorator Pattern） 外观模式（Facade Pattern） 享元模式（Flyweight Pattern） 代理模式（Proxy Pattern）
行为型模式 这些设计模式特别关注对象之间的通信。	责任链模式（Chain of Responsibility Pattern） 命令模式（Command Pattern） 解释器模式（Interpreter Pattern） 迭代器模式（Iterator Pattern） 中介者模式（Mediator Pattern） 备忘录模式（Memento Pattern） 观察者模式（Observer Pattern） 状态模式（State Pattern） 空对象模式（Null Object Pattern） 策略模式（Strategy Pattern） 模板模式（Template Pattern） 访问者模式（Visitor Pattern）
J2EE 模式 这些设计模式特别关注表示层。这些模式是由 Sun Java Center 鉴定的。	MVC 模式（MVC Pattern） 业务代表模式（Business Delegate Pattern） 组合实体模式（Composite Entity Pattern） 数据访问对象模式（Data Access Object Pattern） 前端控制器模式（Front Controller Pattern） 拦截过滤器模式（Intercepting Filter Pattern） 服务定位器模式（Service Locator Pattern） 传输对象模式（Transfer Object Pattern）

下面用一个图片来整体描述一下设计模式之间的关系：

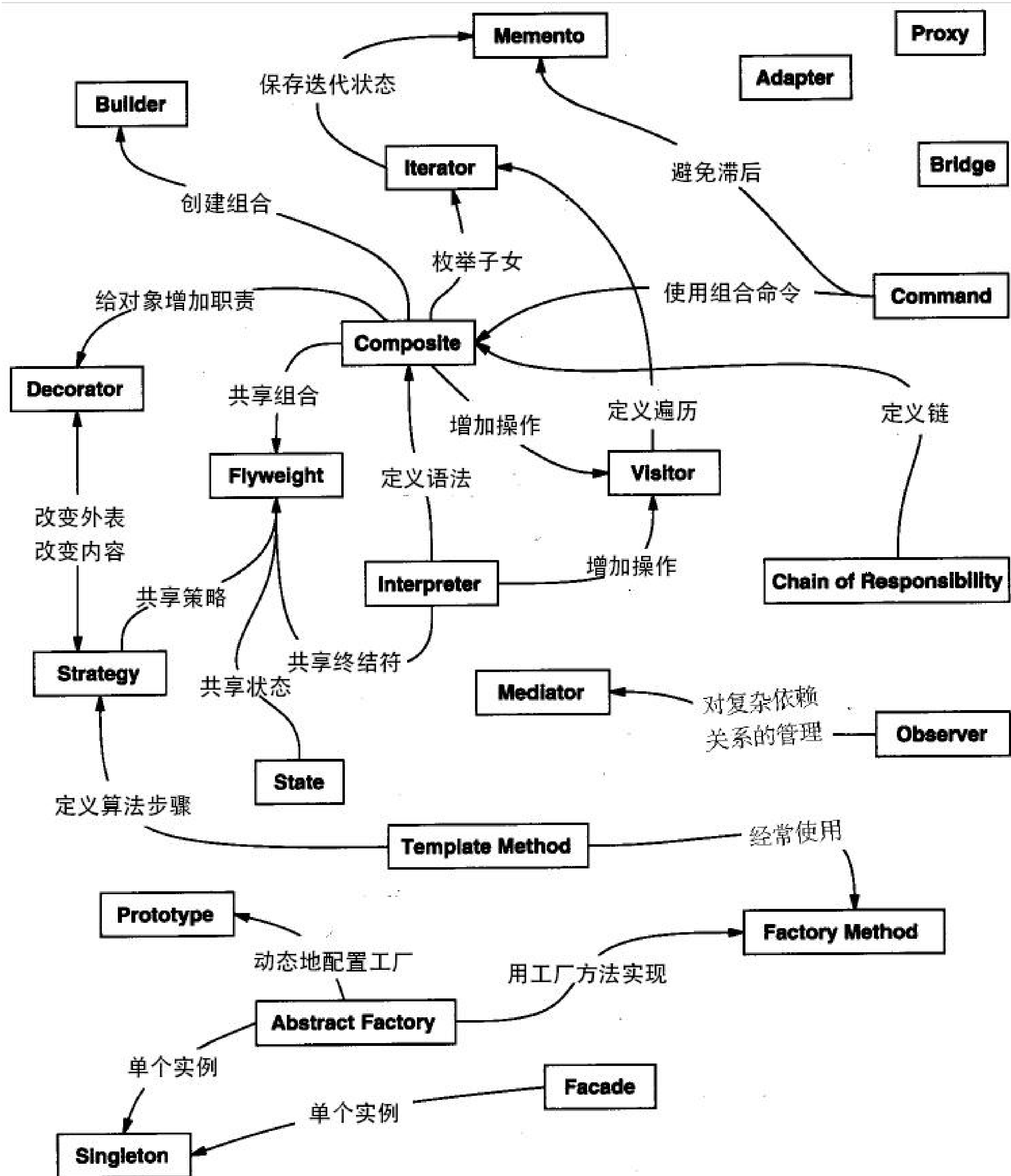


图　设计模式之间的关系

设计模式的六大原则

1、开闭原则（Open Close Principle）

开闭原则的意思是：对扩展开放，对修改关闭。在程序需要进行拓展的时候，不能去修改原有的代码，实现一个热插拔的效果。简言之，是为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类，后面的具体设计中我们会提到这点。

2、里氏代换原则（**Liskov Substitution Principle**）

里氏代换原则是面向对象设计的基本原则之一。 里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。LSP 是继承复用的基石，只有当派生类可以替换掉基类，且软件单位的功能不受到影响时，基类才能真正被复用，而派生类也能够在基类的基础上增加新的行为。里氏代换原则是对开闭原则的补充。实现开闭原则的关键步骤就是抽象化，而基类与子类的继承关系就是抽象化的具体实现，所以里氏代换原则是对实现抽象化的具体步骤的规范。

3、依赖倒转原则（**Dependence Inversion Principle**）

这个原则是开闭原则的基础，具体内容：针对对接口编程，依赖于抽象而不依赖于具体。

4、接口隔离原则（**Interface Segregation Principle**）

这个原则的意思是：使用多个隔离的接口，比使用单个接口要好。它还有另外一个意思是：降低类之间的耦合度。由此可见，其实设计模式就是从大型软件架构出发、便于升级和维护的软件设计思想，它强调降低依赖，降低耦合。

5、迪米特法则，又称最少知道原则（**Demeter Principle**）

最少知道原则是指：一个实体应当尽量少地与其他实体之间发生相互作用，使得系统功能模块相对独立。

6、合成复用原则（**Composite Reuse Principle**）

合成复用原则是指：尽量使用合成/聚合的方式，而不是使用继承。

工厂模式

工厂模式（Factory Pattern）是 Java 中最常用的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在工厂模式中，我们在创建对象时不会对客户端暴露创建逻辑，并且是通过使用一个共同的接口来指向新创建的对象。

介绍

意图：定义一个创建对象的接口，让其子类自己决定实例化哪一个工厂类，工厂模式使其创建过程延迟到子类进行。

主要解决：主要解决接口选择的问题。

何时使用：我们明确地计划不同条件下创建不同实例时。

如何解决：让其子类实现工厂接口，返回的也是一个抽象的产品。

关键代码：创建过程在其子类执行。

应用实例： 1、您需要一辆汽车，可以直接从工厂里面提货，而不用去管这辆汽车是怎么做出来的，以及这个汽车里面的具体实现。 2、Hibernate 换数据库只需换方言和驱动就可以。

优点： 1、一个调用者想创建一个对象，只要知道其名称就可以了。 2、扩展性高，如果想增加一个产品，只要扩展一个工厂类就可以。 3、屏蔽产品的具体实现，调用者只关心产品的接口。

缺点：每次增加一个产品时，都需要增加一个具体类和对象实现工厂，使得系统中类的个数成倍增加，在一定程度上增加了系统的复杂度，同时也增加了系统具体类的依赖。这并不是什么好事。

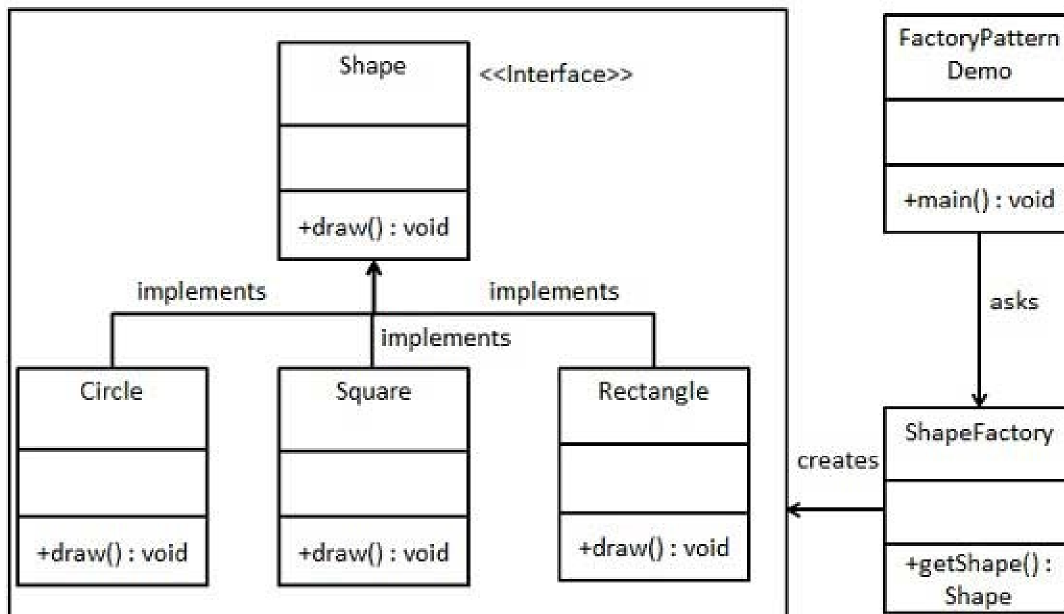
使用场景： 1、日志记录器：记录可能记录到本地硬盘、系统事件、远程服务器等，用户可以选择记录日志到什么地方。 2、数据库访问，当用户不知道最后系统采用哪一类数据库，以及数据库可能有变化时。 3、设计一个连接服务器的框架，需要三个协议，"POP3"、"IMAP"、"HTTP"，可以把这三个作为产品类，共同实现一个接口。

注意事项：作为一种创建类模式，在任何需要生成复杂对象的地方，都可以使用工厂方法模式。有一点需要注意的地方就是复杂对象适合使用工厂模式，而简单对象，特别是只需要通过 new 就可以完成创建的对象，无需使用工厂模式。如果使用工厂模式，就需要引入一个工厂类，会增加系统的复杂度。

实现

我们将创建一个 *Shape* 接口和实现 *Shape* 接口的实体类。下一步是定义工厂类 *ShapeFactory*。

FactoryPatternDemo，我们的演示类使用 *ShapeFactory* 来获取 *Shape* 对象。它将向 *ShapeFactory* 传递信息（*CIRCLE / RECTANGLE / SQUARE*），以便获取它所需对象的类型。



步骤 1

创建一个接口。

Shape.java

```
public interface Shape {
   void draw();
}
```

步骤 2

创建实现接口的实体类。

Rectangle.java

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

Square.java

```
public class Square implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Square::draw() method.");
   }
}
```

Circle.java

```
public class Circle implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Circle::draw() method.");
   }
}
```

步骤 3

创建一个工厂，生成基于给定信息的实体类的对象。

ShapeFactory.java

```
public class ShapeFactory {

   //使用 getShape 方法获取形状类型的对象
   public Shape getShape(String shapeType){
      if(shapeType == null){
         return null;
      }
      if(shapeType.equalsIgnoreCase("CIRCLE")){
         return new Circle();
      } else if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();
      } else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }
      return null;
   }
}
```

步骤 4

使用该工厂，通过传递类型信息来获取实体类的对象。

FactoryPatternDemo.java

```
public class FactoryPatternDemo {

   public static void main(String[] args) {
      ShapeFactory shapeFactory = new ShapeFactory();

      //获取 Circle 的对象，并调用它的 draw 方法
      Shape shape1 = shapeFactory.getShape("CIRCLE");

      //调用 Circle 的 draw 方法
      shape1.draw();

      //获取 Rectangle 的对象，并调用它的 draw 方法
      Shape shape2 = shapeFactory.getShape("RECTANGLE");

      //调用 Rectangle 的 draw 方法
      shape2.draw();

      //获取 Square 的对象，并调用它的 draw 方法
      Shape shape3 = shapeFactory.getShape("SQUARE");

      //调用 Square 的 draw 方法
      shape3.draw();
   }
}
```

步骤 5

验证输出。

```
Inside Circle::draw() method.
Inside Rectangle::draw() method.
Inside Square::draw() method.
```

抽象工厂模式

抽象工厂模式（Abstract Factory Pattern）是围绕一个超级工厂创建其他工厂。该超级工厂又称为其他工厂的工厂。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

在抽象工厂模式中，接口是负责创建一个相关对象的工厂，不需要显式指定它们的类。每个生成的工厂都能按照工厂模式提供对象。

介绍

意图：提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。

主要解决：主要解决接口选择的问题。

何时使用：系统的产品有多于一个的产品族，而系统只消费其中某一族的产品。

如何解决：在一个产品族里面，定义多个产品。

关键代码：在一个工厂里聚合多个同类产品。

应用实例：工作了，为了参加一些聚会，肯定有两套或多套衣服吧，比如说有商务装（成套，一系列具体产品）、时尚装（成套，一系列具体产品），甚至对于一个家庭来说，可能有商务女装、商务男装、时尚女装、时尚男装，这些也都是成套的，即一系列具体产品。假设一种情况（现实中是不存在的，要不然，没法进入共产主义了，但有利于说明抽象工厂模式），在您的家中，某一个衣柜（具体工厂）只能存放某一种这样的衣服（成套，一系列具体产品），每次拿这种成套的衣服时也自然要从这个衣柜中取出了。用 OO 的思想去理解，所有的衣柜（具体工厂）都是衣柜类的（抽象工厂）某一个，而每一件成套的衣服又包括具体的上衣（某一具体产品），裤子（某一具体产品），这些具体的上衣其实也都是上衣（抽象产品），具体的裤子也都是裤子（另一个抽象产品）。

优点：当一个产品族中的多个对象被设计成一起工作时，它能保证客户端始终只使用同一个产品族中的对象。

缺点：产品族扩展非常困难，要增加一个系列的某一产品，既要在抽象的 Creator 里加代码，又要在具体的里面加代码。

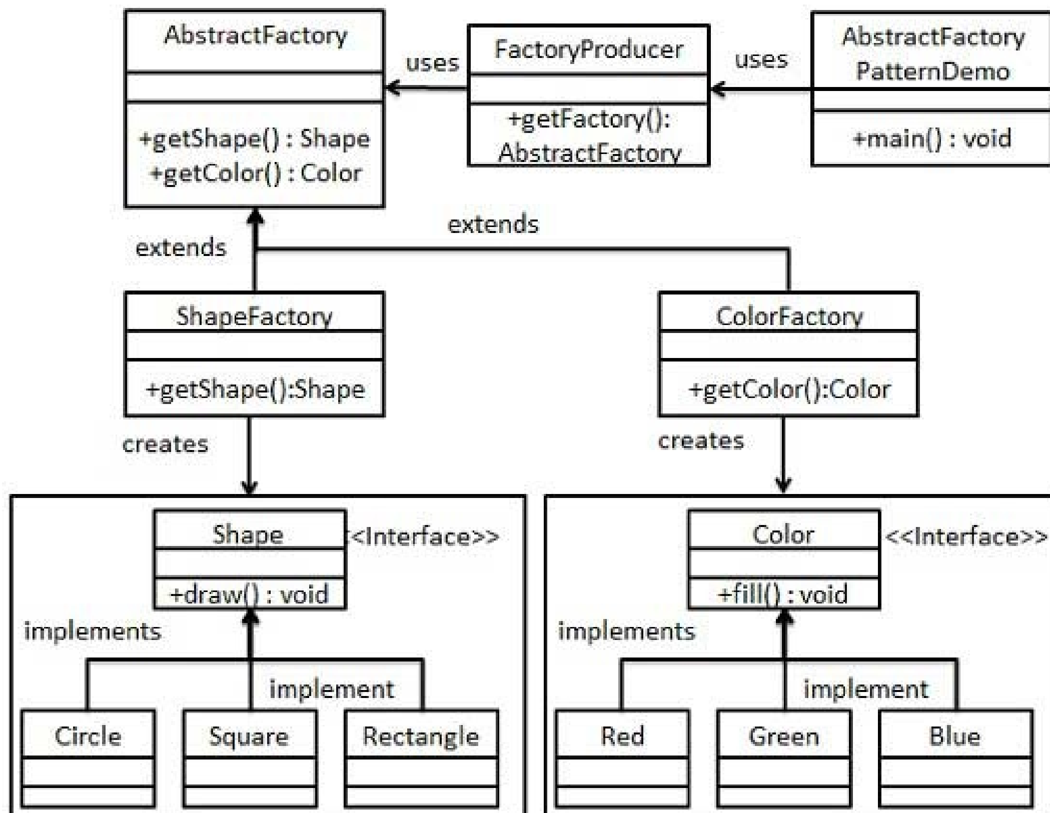
使用场景： 1、QQ 换皮肤，一整套一起换。 2、生成不同操作系统的程序。

注意事项：产品族难扩展，产品等级易扩展。

实现

我们将创建 *Shape* 和 *Color* 接口和实现这些接口的实体类。下一步是创建抽象工厂类 *AbstractFactory*。接着定义工厂类 *ShapeFactory* 和 *ColorFactory*，这两个工厂类都是扩展了 *AbstractFactory*。然后创建一个工厂创造器/生成器类 *FactoryProducer*。

AbstractFactoryPatternDemo，我们的演示类使用 *FactoryProducer* 来获取 *AbstractFactory* 对象。它将向 *AbstractFactory* 传递形状信息 *Shape*（*CIRCLE / RECTANGLE / SQUARE*），以便获取它所需对象的类型。同时它还向 *AbstractFactory* 传递颜色信息 *Color*（*RED / GREEN / BLUE*），以便获取它所需对象的类型。



步骤 1

为形状创建一个接口。

Shape.java

```
public interface Shape {
   void draw();
}
```

步骤 2

创建实现接口的实体类。

Rectangle.java

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

Square.java

```
public class Square implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Square::draw() method.");
   }
}
```

Circle.java

```
public class Circle implements Shape {

   @Override
   public void draw() {
      System.out.println("Inside Circle::draw() method.");
   }
}
```

步骤 3

为颜色创建一个接口。

Color.java

```
public interface Color {
   void fill();
}
```

步骤4

创建实现接口的实体类。

Red.java

```
public class Red implements Color {

   @Override
   public void fill() {
      System.out.println("Inside Red::fill() method.");
   }
}
```

Green.java

```
public class Green implements Color {

   @Override
   public void fill() {
      System.out.println("Inside Green::fill() method.");
   }
}
```

Blue.java

```
public class Blue implements Color {

   @Override
   public void fill() {
      System.out.println("Inside Blue::fill() method.");
   }
}
```

步骤 5

为 Color 和 Shape 对象创建抽象类来获取工厂。

AbstractFactory.java

```
public abstract class AbstractFactory {
   abstract Color getColor(String color);
   abstract Shape getShape(String shape) ;
}
```

步骤 6

创建扩展了 AbstractFactory 的工厂类，基于给定的信息生成实体类的对象。

ShapeFactory.java

```
public class ShapeFactory extends AbstractFactory {

   @Override
   public Shape getShape(String shapeType){
      if(shapeType == null){
         return null;
      }
      if(shapeType.equalsIgnoreCase("CIRCLE")){
         return new Circle();
      } else if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();
      } else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }
      return null;
   }

   @Override
   Color getColor(String color) {
      return null;
   }
}
```

ColorFactory.java

```
public class ColorFactory extends AbstractFactory {

   @Override
   public Shape getShape(String shapeType){
      return null;
   }

   @Override
   Color getColor(String color) {
      if(color == null){
         return null;
      }
      if(color.equalsIgnoreCase("RED")){
         return new Red();
      } else if(color.equalsIgnoreCase("GREEN")){
         return new Green();
      } else if(color.equalsIgnoreCase("BLUE")){
         return new Blue();
      }
      return null;
   }
}
```

步骤 7

创建一个工厂创造器/生成器类，通过传递形状或颜色信息来获取工厂。

FactoryProducer.java

```
public class FactoryProducer {
   public static AbstractFactory getFactory(String choice){
      if(choice.equalsIgnoreCase("SHAPE")){
         return new ShapeFactory();
      } else if(choice.equalsIgnoreCase("COLOR")){
         return new ColorFactory();
      }
      return null;
   }
}
```

步骤 8

使用 FactoryProducer 来获取 AbstractFactory，通过传递类型信息来获取实体类的对象。

AbstractFactoryPatternDemo.java

```
public class AbstractFactoryPatternDemo {
   public static void main(String[] args) {

      //获取形状工厂
      AbstractFactory shapeFactory = FactoryProducer.getFactory("SHAPE");

      //获取形状为 Circle 的对象
      Shape shape1 = shapeFactory.getShape("CIRCLE");

      //调用 Circle 的 draw 方法
      shape1.draw();

      //获取形状为 Rectangle 的对象
      Shape shape2 = shapeFactory.getShape("RECTANGLE");

      //调用 Rectangle 的 draw 方法
      shape2.draw();

      //获取形状为 Square 的对象
      Shape shape3 = shapeFactory.getShape("SQUARE");

      //调用 Square 的 draw 方法
      shape3.draw();

      //获取颜色工厂
      AbstractFactory colorFactory = FactoryProducer.getFactory("COLOR");

      //获取颜色为 Red 的对象
      Color color1 = colorFactory.getColor("RED");

      //调用 Red 的 fill 方法
      color1.fill();

      //获取颜色为 Green 的对象
      Color color2 = colorFactory.getColor("Green");

      //调用 Green 的 fill 方法
      color2.fill();

      //获取颜色为 Blue 的对象
      Color color3 = colorFactory.getColor("BLUE");

      //调用 Blue 的 fill 方法
      color3.fill();
   }
}
```

步骤 9

验证输出。

```
Inside Circle::draw() method.
Inside Rectangle::draw() method.
Inside Square::draw() method.
Inside Red::fill() method.
Inside Green::fill() method.
Inside Blue::fill() method.
```

单例模式

单例模式（Singleton Pattern）是 Java 中最简单的设计模式之一。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

这种模式涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。这个类提供了一种访问其唯一的对象的方式，可以直接访问，不需要实例化该类的对象。

注意：

- 1、单例类只能有一个实例。
- 2、单例类必须自己创建自己的唯一实例。
- 3、单例类必须给所有其他对象提供这一实例。

介绍

意图：保证一个类仅有一个实例，并提供一个访问它的全局访问点。

主要解决：一个全局使用的类频繁地创建与销毁。

何时使用：当您想控制实例数目，节省系统资源的时候。

如何解决：判断系统是否已经有这个单例，如果有则返回，如果没有则创建。

关键代码：构造函数是私有的。

应用实例： 1、一个党只能有一个主席。 2、Windows 是多进程多线程的，在操作一个文件的时候，就不可避免地出现多个进程或线程同时操作一个文件的现象，所以所有文件的处理必须通过唯一的实例来进行。 3、一些设备管理器常常设计为单例模式，比如一个电脑有两台打印机，在输出的时候就要处理不能两台打印机打印同一个文件。

优点： 1、在内存里只有一个实例，减少了内存的开销，尤其是频繁的创建和销毁实例（比如管理学院首页页面缓存）。 2、避免对资源的多重占用（比如写文件操作）。

缺点：没有接口，不能继承，与单一职责原则冲突，一个类应该只关心内部逻辑，而不关心外面怎么样来实例化。

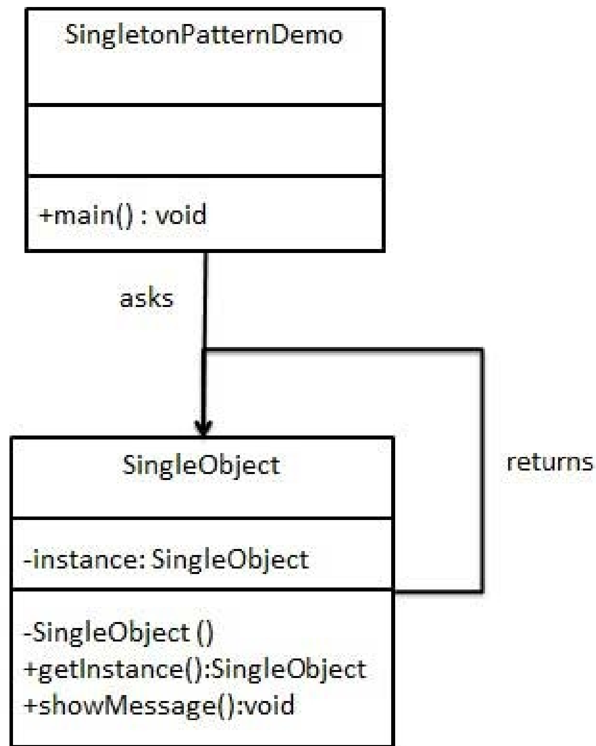
使用场景： 1、要求生产唯一序列号。 2、WEB 中的计数器，不用每次刷新都在数据库里加一次，用单例先缓存起来。 3、创建的一个对象需要消耗的资源过多，比如 I/O 与数据库的连接等。

注意事项：getInstance() 方法中需要使用同步锁 synchronized (Singleton.class) 防止多线程同时进入造成 instance 被多次实例化。

实现

我们将创建一个 *SingleObject* 类。*SingleObject* 类有它的私有构造函数和本身的一个静态实例。

SingleObject 类提供了一个静态方法，供外界获取它的静态实例。*SingletonPatternDemo*，我们的演示类使用 *SingleObject* 类来获取 *SingleObject* 对象。



步骤 1

创建一个 Singleton 类。

SingleObject.java

```
public class SingleObject {

   //创建 SingleObject 的一个对象
   private static SingleObject instance = new SingleObject();

   //让构造函数为 private，这样该类就不会被实例化
   private SingleObject(){}

   //获取唯一可用的对象
   public static SingleObject getInstance(){
      return instance;
   }

   public void showMessage(){
      System.out.println("Hello World!");
   }
}
```

步骤 2

从 singleton 类获取唯一的对象。

SingletonPatternDemo.java

```
public class SingletonPatternDemo {
   public static void main(String[] args) {

      //不合法的构造函数
      //编译时错误：构造函数 SingleObject() 是不可见的
      //SingleObject object = new SingleObject();

      //获取唯一可用的对象
      SingleObject object = SingleObject.getInstance();

      //显示消息
      object.showMessage();
   }
}
```

步骤 3

验证输出。

```
Hello World!
```

单例模式的几种实现方式

单例模式的实现有多种方式，如下所示：

1、懒汉式，线程不安全

是否 **Lazy** 初始化：是

是否多线程安全：否

实现难度：易

描述：这种方式是最基本的实现方式，这种实现最大的问题就是不支持多线程。因为没有加锁 synchronized，所以严格意义上它并不算单例模式。
这种方式 lazy loading 很明显，不要求线程安全，在多线程不能正常工作。

代码实例：

```
public class Singleton {
    private static Singleton instance;
    private Singleton (){}

    public static Singleton getInstance() {
    if (instance == null) {
        instance = new Singleton();
    }
    return instance;
    }
}
```

接下来介绍的几种实现方式都支持多线程，但是在性能上有所差异。

2、懒汉式，线程安全

是否 **Lazy** 初始化：是

是否多线程安全：是

实现难度：易

描述：这种方式具备很好的 lazy loading，能够在多线程中很好的工作，但是，效率很低，99% 情况下不需要同步。
优点：第一次调用才初始化，避免内存浪费。
缺点：必须加锁 synchronized 才能保证单例，但加锁会影响效率。
getInstance() 的性能对应用程序不是很关键（该方法使用不太频繁）。

代码实例：

```
public class Singleton {
    private static Singleton instance;
    private Singleton (){}
    public static synchronized Singleton getInstance() {
    if (instance == null) {
        instance = new Singleton();
    }
    return instance;
    }
}
```

3、饿汉式

是否 **Lazy** 初始化：否

是否多线程安全：是

实现难度：易

描述：这种方式比较常用，但容易产生垃圾对象。
优点：没有加锁，执行效率会提高。
缺点：类加载时就初始化，浪费内存。

它基于 classloder 机制避免了多线程的同步问题，不过，instance 在类装载时就实例化，虽然导致类装载的原因有很多种，在单例模式中大多数都是调用 getInstance 方法， 但是也不能确定有其他的方式（或者其他的静态方法）导致类装载，这时候初始化 instance 显然没有达到 lazy loading 的效果。

代码实例：

```
public class Singleton {
    private static Singleton instance = new Singleton();
    private Singleton (){}
    public static Singleton getInstance() {
    return instance;
    }
}
```

4、双检锁/双重校验锁（DCL，即 double-checked locking）

JDK 版本：JDK1.5 起

是否 **Lazy** 初始化：是

是否多线程安全：是

实现难度：较复杂

描述：这种方式采用双锁机制，安全且在多线程情况下能保持高性能。
getInstance() 的性能对应用程序很关键。

代码实例：

```
public class Singleton {
    private volatile static Singleton singleton;
    private Singleton (){}
    public static Singleton getSingleton() {
    if (singleton == null) {
        synchronized (Singleton.class) {
        if (singleton == null) {
            singleton = new Singleton();
        }
        }
    }
    return singleton;
    }
}
```

5、登记式/静态内部类

是否 **Lazy** 初始化：是

是否多线程安全：是

实现难度：一般

描述：这种方式能达到双检锁方式一样的功效，但实现更简单。对静态域使用延迟初始化，应使用这种方式而不是双检锁方式。这种方式只适用于静态域的情况，双检锁方式可在实例域需要延迟初始化时使用。
这种方式同样利用了 classloder 机制来保证初始化 instance 时只有一个线程，它跟第 3 种方式不同的是：第 3 种方式只要 Singleton 类被装载了，那么 instance 就会被实例化（没有达到 lazy loading 效果），而这种方式是 Singleton 类被装载了，instance 不一定被初始化。因为 SingletonHolder 类没有被主动使用，只有显示通过调用 getInstance 方法时，才会显示装载 SingletonHolder 类，从而实例化 instance。想象一下，如果实例化 instance 很消耗资源，所以想让它延迟加载，另外一方面，又不希望在 Singleton 类加载时就实例化，因为不能确保 Singleton 类还可能在其他的地方被主动使用从而被加载，那么这个时候实例化 instance 显然是不合适的。这个时候，这种方式相比第 3 种方式就显得很合理。

代码实例：

```
public class Singleton {
    private static class SingletonHolder {
    private static final Singleton INSTANCE = new Singleton();
    }
    private Singleton (){}
    public static final Singleton getInstance() {
    return SingletonHolder.INSTANCE;
    }
}
```

6、枚举

JDK 版本：JDK1.5 起

是否 **Lazy** 初始化：否

是否多线程安全：是

实现难度：易

描述：这种实现方式还没有被广泛采用，但这是实现单例模式的最佳方法。它更简洁，自动支持序列化机制，绝对防止多次实例化。
这种方式是 Effective Java 作者 Josh Bloch 提倡的方式，它不仅能避免多线程同步问题，而且还自动支持序列化机制，防止反序列化重新创建新的对象，绝对防止多次实例化。不过，由于 JDK1.5 之后才加入 enum 特性，用这种方式写不免让人感觉生疏，在实际工作中，也很少用。
不能通过 reflection attack 来调用私有构造方法。

代码实例：

```
public enum Singleton {
    INSTANCE;
    public void whateverMethod() {
    }
}
```

经验之谈：一般情况下，不建议使用第 1 种和第 2 种懒汉方式，建议使用第 3 种饿汉方式。只有在要明确实现 lazy loading 效果时，才会使用第 5 种登记方式。如果涉及到反序列化创建对象时，可以尝试使用第 6 种枚举方式。如果有其他特殊的需求，可以考虑使用第 4 种双检锁方式。

建造者模式

建造者模式（Builder Pattern）使用多个简单的对象一步一步构建成一个复杂的对象。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

一个 Builder 类会一步一步构造最终的对象。该 Builder 类是独立于其他对象的。

介绍

意图：将一个复杂的构建与其表示相分离，使得同样的构建过程可以创建不同的表示。

主要解决：主要解决在软件系统中，有时候面临着"一个复杂对象"的创建工作，其通常由各个部分的子对象用一定的算法构成；由于需求的变化，这个复杂对象的各个部分经常面临着剧烈的变化，但是将它们组合在一起的算法却相对稳定。

何时使用：一些基本部件不会变，而其组合经常变化的时候。

如何解决：将变与不变分离开。

关键代码：建造者：创建和提供实例，导演：管理建造出来的实例的依赖关系。

应用实例： 1、去肯德基，汉堡、可乐、薯条、炸鸡翅等是不变的，而其组合是经常变化的，生成出所谓的"套餐"。 2、JAVA 中的 StringBuilder。

优点： 1、建造者独立，易扩展。 2、便于控制细节风险。

缺点： 1、产品必须有共同点，范围有限制。 2、如内部变化复杂，会有很多的建造类。

使用场景： 1、需要生成的对象具有复杂的内部结构。 2、需要生成的对象内部属性本身相互依赖。

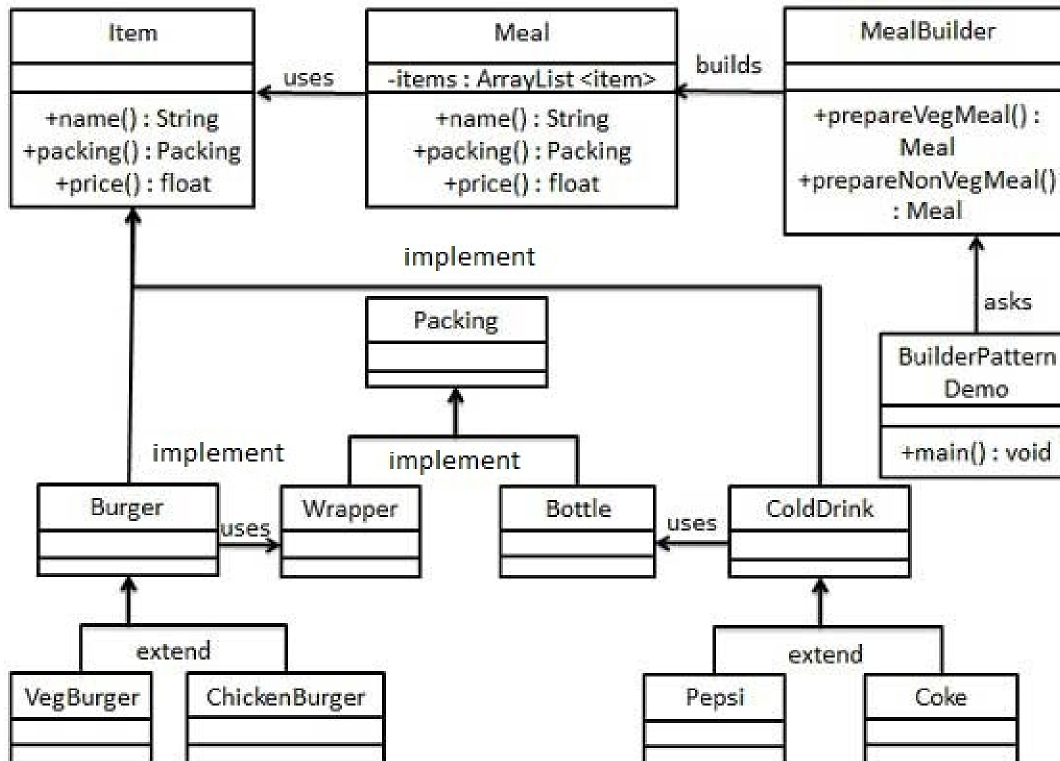
注意事项：与工厂模式的区别是：建造者模式更加关注与零件装配的顺序。

实现

我们假设一个快餐店的商业案例，其中，一个典型的套餐可以是一个汉堡（Burger）和一杯冷饮（Cold drink）。汉堡（Burger）可以是素食汉堡（Veg Burger）或鸡肉汉堡（Chicken Burger），它们是包在纸盒中。冷饮（Cold drink）可以是可口可乐（coke）或百事可乐（pepsi），它们是装在瓶子中。

我们将创建一个表示食物条目（比如汉堡和冷饮）的 *Item* 接口和实现 *Item* 接口的实体类，以及一个表示食物包装的 *Packing* 接口和实现 *Packing* 接口的实体类，汉堡是包在纸盒中，冷饮是装在瓶子中。

然后我们创建一个 *Meal* 类，带有 *Item* 的 *ArrayList* 和一个通过结合 *Item* 来创建不同类型的 *Meal* 对象的 *MealBuilder*。*BuilderPatternDemo*，我们的演示类使用 *MealBuilder* 来创建一个 *Meal*。



步骤 1

创建一个表示食物条目和食物包装的接口。

Item.java

```
public interface Item {
   public String name();
   public Packing packing();
   public float price();
}
```

Packing.java

```
public interface Packing {
   public String pack();
}
```

步骤 2

创建实现 Packing 接口的实体类。

Wrapper.java

```
public class Wrapper implements Packing {

   @Override
   public String pack() {
      return "Wrapper";
   }
}
```

Bottle.java

```
public class Bottle implements Packing {

   @Override
   public String pack() {
      return "Bottle";
   }
}
```

步骤 3

创建实现 Item 接口的抽象类，该类提供了默认的功能。

Burger.java

```
public abstract class Burger implements Item {

   @Override
   public Packing packing() {
      return new Wrapper();
   }

   @Override
   public abstract float price();
}
```

ColdDrink.java

```
public abstract class ColdDrink implements Item {

    @Override
    public Packing packing() {
       return new Bottle();
    }

    @Override
    public abstract float price();
}
```

步骤 4

创建扩展了 Burger 和 ColdDrink 的实体类。

VegBurger.java

```
public class VegBurger extends Burger {

   @Override
   public float price() {
      return 25.0f;
   }

   @Override
   public String name() {
      return "Veg Burger";
   }
}
```

ChickenBurger.java

```
public class ChickenBurger extends Burger {

   @Override
   public float price() {
      return 50.5f;
   }

   @Override
   public String name() {
      return "Chicken Burger";
   }
}
```

Coke.java

```
public class Coke extends ColdDrink {

   @Override
   public float price() {
      return 30.0f;
   }

   @Override
   public String name() {
      return "Coke";
   }
}
```

Pepsi.java

```
public class Pepsi extends ColdDrink {

   @Override
   public float price() {
      return 35.0f;
   }

   @Override
   public String name() {
      return "Pepsi";
   }
}
```

步骤 5

创建一个 Meal 类，带有上面定义的 Item 对象。

Meal.java

```
import java.util.ArrayList;
import java.util.List;

public class Meal {
   private List<Item> items = new ArrayList<Item>();

   public void addItem(Item item){
      items.add(item);
   }

   public float getCost(){
      float cost = 0.0f;
      for (Item item : items) {
         cost += item.price();
      }
      return cost;
   }

   public void showItems(){
      for (Item item : items) {
         System.out.print("Item : "+item.name());
         System.out.print(", Packing : "+item.packing().pack());
         System.out.println(", Price : "+item.price());
      }
   }
}
```

步骤 6

创建一个 MealBuilder 类，实际的 builder 类负责创建 Meal 对象。

MealBuilder.java

```
public class MealBuilder {

   public Meal prepareVegMeal (){
      Meal meal = new Meal();
      meal.addItem(new VegBurger());
      meal.addItem(new Coke());
      return meal;
   }

   public Meal prepareNonVegMeal (){
      Meal meal = new Meal();
      meal.addItem(new ChickenBurger());
      meal.addItem(new Pepsi());
      return meal;
   }
}
```

步骤 7

BuiderPatternDemo 使用 MealBuider 来演示建造者模式（Builder Pattern）。

BuilderPatternDemo.java

```
public class BuilderPatternDemo {
   public static void main(String[] args) {
      MealBuilder mealBuilder = new MealBuilder();

      Meal vegMeal = mealBuilder.prepareVegMeal();
      System.out.println("Veg Meal");
      vegMeal.showItems();
      System.out.println("Total Cost: " +vegMeal.getCost());

      Meal nonVegMeal = mealBuilder.prepareNonVegMeal();
      System.out.println("\n\nNon-Veg Meal");
      nonVegMeal.showItems();
      System.out.println("Total Cost: " +nonVegMeal.getCost());
   }
}
```

步骤 8

验证输出。

```
Veg Meal
Item : Veg Burger, Packing : Wrapper, Price : 25.0
Item : Coke, Packing : Bottle, Price : 30.0
Total Cost: 55.0

Non-Veg Meal
Item : Chicken Burger, Packing : Wrapper, Price : 50.5
Item : Pepsi, Packing : Bottle, Price : 35.0
Total Cost: 85.5
```

原型模式

原型模式（Prototype Pattern）是用于创建重复的对象，同时又能保证性能。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。

这种模式是实现了一个原型接口，该接口用于创建当前对象的克隆。当直接创建对象的代价比较大时，则采用这种模式。例如，一个对象需要在一个高代价的数据库操作之后被创建。我们可以缓存该对象，在下一个请求时返回它的克隆，在需要的时候更新数据库，以此来减少数据库调用。

介绍

意图：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。

主要解决：在运行期建立和删除原型。

何时使用： 1、当一个系统应该独立于它的产品创建，构成和表示时。 2、当要实例化的类是在运行时刻指定时，例如，通过动态装载。 3、为了避免创建一个与产品类层次平行的工厂类层次时。 4、当一个类的实例只能有几个不同状态组合中的一种时。建立相应数目的原型并克隆它们可能比每次用合适的状态手工实例化该类更方便一些。

如何解决：利用已有的一个原型对象，快速地生成和原型对象一样的实例。

关键代码： 1、实现克隆操作，在 JAVA 继承 Cloneable，重写 clone()，在 .NET 中可以使用 Object 类的 MemberwiseClone() 方法来实现对象的浅拷贝或通过序列化的方式来实现深拷贝。 2、原型模式同样用于隔离类对象的使用者和具体类型（易变类）之间的耦合关系，它同样要求这些"易变类"拥有稳定的接口。

应用实例： 1、细胞分裂。 2、JAVA 中的 Object clone() 方法。

优点： 1、性能提高。 2、逃避构造函数的约束。

缺点： 1、配备克隆方法需要对类的功能进行通盘考虑，这对于全新的类不是很难，但对于已有的类不一定很容易，特别当一个类引用不支持串行化的间接对象，或者引用含有循环结构的时候。 2、必须实现 Cloneable 接口。 3、逃避构造函数的约束。

使用场景： 1、资源优化场景。 2、类初始化需要消化非常多的资源，这个资源包括数据、硬件资源等。 3、性能和安全要求的场景。 4、通过 new 产生一个对象需要非常繁琐的数据准备或访问权限，则可以使用原型模式。 5、一个对象多个修改者的场景。 6、一个对象需要提供给其他对象访问，而且各个调用者可能都需要修改其值时，可以考虑使用原型模式拷贝多

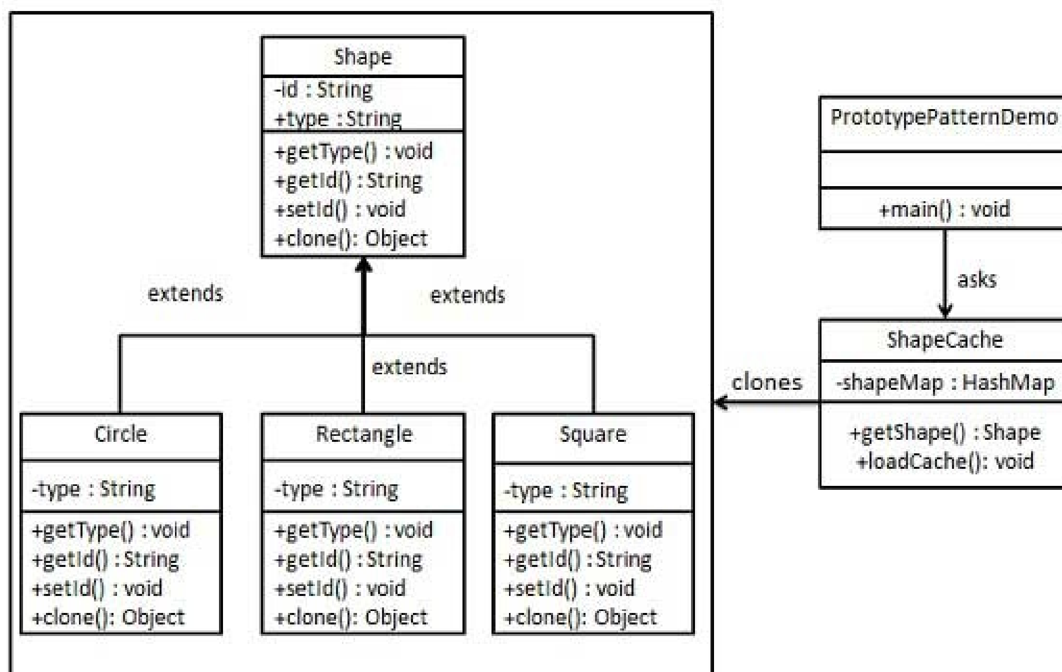
个对象供调用者使用。 7、在实际项目中，原型模式很少单独出现，一般是和工厂方法模式一起出现，通过 clone 的方法创建一个对象，然后由工厂方法提供给调用者。原型模式已经与 Java 融为浑然一体，大家可以随手拿来使用。

注意事项：与通过对一个类进行实例化来构造新对象不同的是，原型模式是通过拷贝一个现有对象生成新对象的。浅拷贝实现 Cloneable，重写，深拷贝是通过实现 Serializable 读取二进制流。

实现

我们将创建一个抽象类 *Shape* 和扩展了 *Shape* 类的实体类。下一步是定义类 *ShapeCache*，该类把 shape 对象存储在一个 *Hashtable* 中，并在请求的时候返回它们的克隆。

PrototypPatternDemo，我们的演示类使用 *ShapeCache* 类来获取 *Shape* 对象。



步骤 1

创建一个实现了 *Clonable* 接口的抽象类。

Shape.java

```
public abstract class Shape implements Cloneable {

   private String id;
   protected String type;

   abstract void draw();

   public String getType(){
      return type;
   }

   public String getId() {
      return id;
   }

   public void setId(String id) {
      this.id = id;
   }

   public Object clone() {
      Object clone = null;
      try {
         clone = super.clone();
      } catch (CloneNotSupportedException e) {
         e.printStackTrace();
      }
      return clone;
   }
}
```

步骤 2

创建扩展了上面抽象类的实体类。

Rectangle.java

```
public class Rectangle extends Shape {

   public Rectangle(){
     type = "Rectangle";
   }

   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

Square.java

```
public class Square extends Shape {

   public Square(){
     type = "Square";
   }

   @Override
   public void draw() {
      System.out.println("Inside Square::draw() method.");
   }
}
```

Circle.java

```
public class Circle extends Shape {

   public Circle(){
     type = "Circle";
   }

   @Override
   public void draw() {
      System.out.println("Inside Circle::draw() method.");
   }
}
```

步骤 3

创建一个类，从数据库获取实体类，并把它们存储在一个 *Hashtable* 中。

ShapeCache.java

```
import java.util.Hashtable;

public class ShapeCache {

   private static Hashtable<String, Shape> shapeMap
      = new Hashtable<String, Shape>();

   public static Shape getShape(String shapeId) {
      Shape cachedShape = shapeMap.get(shapeId);
      return (Shape) cachedShape.clone();
   }

   // 对每种形状都运行数据库查询，并创建该形状
   // shapeMap.put(shapeKey, shape);
   // 例如，我们要添加三种形状
   public static void loadCache() {
      Circle circle = new Circle();
      circle.setId("1");
      shapeMap.put(circle.getId(),circle);

      Square square = new Square();
      square.setId("2");
      shapeMap.put(square.getId(),square);

      Rectangle rectangle = new Rectangle();
      rectangle.setId("3");
      shapeMap.put(rectangle.getId(),rectangle);
   }
}
```

步骤 4

PrototypePatternDemo 使用 *ShapeCache* 类来获取存储在 *Hashtable* 中的形状的克隆。

PrototypePatternDemo.java

```
public class PrototypePatternDemo {
   public static void main(String[] args) {
      ShapeCache.loadCache();

      Shape clonedShape = (Shape) ShapeCache.getShape("1");
      System.out.println("Shape : " + clonedShape.getType());

      Shape clonedShape2 = (Shape) ShapeCache.getShape("2");
      System.out.println("Shape : " + clonedShape2.getType());

      Shape clonedShape3 = (Shape) ShapeCache.getShape("3");
      System.out.println("Shape : " + clonedShape3.getType());
   }
}
```

步骤 5

验证输出。

```
Shape : Circle
Shape : Square
Shape : Rectangle
```

适配器模式

适配器模式（Adapter Pattern）是作为两个不兼容的接口之间的桥梁。这种类型的设计模式属于结构型模式，它结合了两个独立接口的功能。

这种模式涉及到一个单一的类，该类负责加入独立的或不兼容的接口功能。举个真实的例子，读卡器是作为内存卡和笔记本之间的适配器。您将内存卡插入读卡器，再将读卡器插入笔记本，这样就可以通过笔记本来读取内存卡。

我们通过下面的实例来演示适配器模式的使用。其中，音频播放器设备只能播放 mp3 文件，通过使用一个更高级的音频播放器来播放 vlc 和 mp4 文件。

介绍

意图：将一个类的接口转换成客户希望的另外一个接口。适配器模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

主要解决：主要解决在软件系统中，常常要将一些"现存的对象"放到新的环境中，而新环境要求的接口是现对象不能满足的。

何时使用： 1、系统需要使用现有的类，而此类的接口不符合系统的需要。 2、想要建立一个可以重复使用的类，用于与一些彼此之间没有太大关联的一些类，包括一些可能在将来引进的类一起工作，这些源类不一定有一致的接口。 3、通过接口转换，将一个类插入另一个类系中。（比如老虎和飞禽，现在多了一个飞虎，在不增加实体的需求下，增加一个适配器，在里面包容一个虎对象，实现飞的接口。）

如何解决：继承或依赖（推荐）。

关键代码：适配器继承或依赖已有的对象，实现想要的目标接口。

应用实例： 1、美国电器 110V，中国 220V，就要有一个适配器将 110V 转化为 220V。 2、JAVA JDK 1.1 提供了 Enumeration 接口，而在 1.2 中提供了 Iterator 接口，想要使用 1.2 的 JDK，则要将以前系统的 Enumeration 接口转化为 Iterator 接口，这时就需要适配器模式。 3、在 LINUX 上运行 WINDOWS 程序。 4、JAVA 中的 jdbc。

优点： 1、可以让任何两个没有关联的类一起运行。 2、提高了类的复用。 3、增加了类的透明度。 4、灵活性好。

缺点： 1、过多地使用适配器，会让系统非常零乱，不易整体进行把握。比如，明明看到调用的是 A 接口，其实内部被适配成了 B 接口的实现，一个系统如果太多出现这种情况，无异于一场灾难。因此如果不是很有必要，可以不使用适配器，而是直接对系统进行重构。 2.由于 JAVA 至多继承一个类，所以至多只能适配一个适配者类，而且目标类必须是抽象类。

使用场景：有动机地修改一个正常运行的系统的接口，这时应该考虑使用适配器模式。

注意事项：适配器不是在详细设计时添加的，而是解决正在服役的项目的问题。

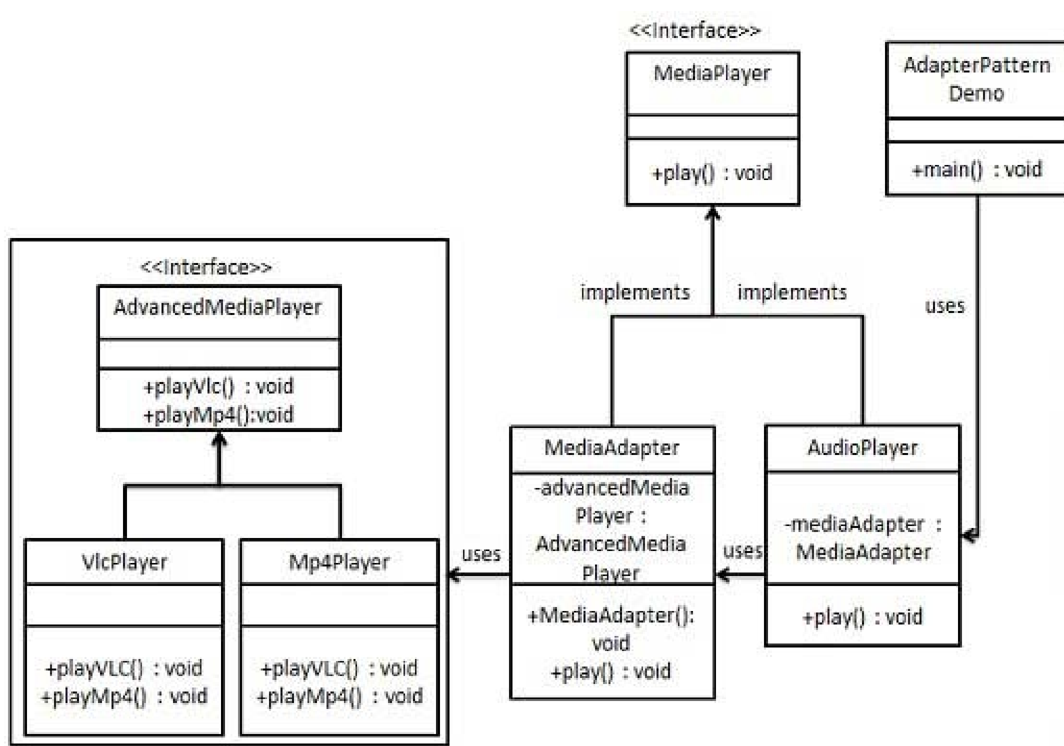
实现

我们有一个 *MediaPlayer* 接口和一个实现了 *MediaPlayer* 接口的实体类 *AudioPlayer*。默认情况下，*AudioPlayer* 可以播放 mp3 格式的音频文件。

我们还有另一个接口 *AdvancedMediaPlayer* 和实现了 *AdvancedMediaPlayer* 接口的实体类。该类可以播放 vlc 和 mp4 格式的文件。

我们想要让 *AudioPlayer* 播放其他格式的音频文件。为了实现这个功能，我们需要创建一个实现了 *MediaPlayer* 接口的适配器类 *MediaAdapter*，并使用 *AdvancedMediaPlayer* 对象来播放所需的格式。

AudioPlayer 使用适配器类 *MediaAdapter* 传递所需的音频类型，不需要知道能播放所需格式音频的实际类。*AdapterPatternDemo*，我们的演示类使用 *AudioPlayer* 类来播放各种格式。



步骤 1

为媒体播放器和更高级的媒体播放器创建接口。

MediaPlayer.java

```
public interface MediaPlayer {
   public void play(String audioType, String fileName);
}
```

AdvancedMediaPlayer.java

```
public interface AdvancedMediaPlayer {
   public void playVlc(String fileName);
   public void playMp4(String fileName);
}
```

步骤 2

创建实现了 *AdvancedMediaPlayer* 接口的实体类。

VlcPlayer.java

```
public class VlcPlayer implements AdvancedMediaPlayer{
   @Override
   public void playVlc(String fileName) {
      System.out.println("Playing vlc file. Name: "+ fileName);
   }

   @Override
   public void playMp4(String fileName) {
      //什么也不做
   }
}
```

Mp4Player.java

```
public class Mp4Player implements AdvancedMediaPlayer{

   @Override
   public void playVlc(String fileName) {
      //什么也不做
   }

   @Override
   public void playMp4(String fileName) {
      System.out.println("Playing mp4 file. Name: "+ fileName);
   }
}
```

步骤 3

创建实现了 *MediaPlayer* 接口的适配器类。

MediaAdapter.java

```
public class MediaAdapter implements MediaPlayer {

   AdvancedMediaPlayer advancedMusicPlayer;

   public MediaAdapter(String audioType){
      if(audioType.equalsIgnoreCase("vlc") ){
         advancedMusicPlayer = new VlcPlayer();
      } else if (audioType.equalsIgnoreCase("mp4")){
         advancedMusicPlayer = new Mp4Player();
      }
   }

   @Override
   public void play(String audioType, String fileName) {
      if(audioType.equalsIgnoreCase("vlc")){
         advancedMusicPlayer.playVlc(fileName);
      }else if(audioType.equalsIgnoreCase("mp4")){
         advancedMusicPlayer.playMp4(fileName);
      }
   }
}
```

步骤 4

创建实现了 *MediaPlayer* 接口的实体类。

AudioPlayer.java

```
public class AudioPlayer implements MediaPlayer {
   MediaAdapter mediaAdapter;

   @Override
   public void play(String audioType, String fileName) {

      //播放 mp3 音乐文件的内置支持
      if(audioType.equalsIgnoreCase("mp3")){
         System.out.println("Playing mp3 file. Name: "+ fileName);
      }
      //mediaAdapter 提供了播放其他文件格式的支持
      else if(audioType.equalsIgnoreCase("vlc")
         || audioType.equalsIgnoreCase("mp4")){
         mediaAdapter = new MediaAdapter(audioType);
         mediaAdapter.play(audioType, fileName);
      }
      else{
         System.out.println("Invalid media. "+
            audioType + " format not supported");
      }
   }
}
```

步骤 5

使用 AudioPlayer 来播放不同类型的音频格式。

AdapterPatternDemo.java

```
public class AdapterPatternDemo {
   public static void main(String[] args) {
      AudioPlayer audioPlayer = new AudioPlayer();

      audioPlayer.play("mp3", "beyond the horizon.mp3");
      audioPlayer.play("mp4", "alone.mp4");
      audioPlayer.play("vlc", "far far away.vlc");
      audioPlayer.play("avi", "mind me.avi");
   }
}
```

步骤 6

验证输出。

```
Playing mp3 file. Name: beyond the horizon.mp3
Playing mp4 file. Name: alone.mp4
Playing vlc file. Name: far far away.vlc
Invalid media. avi format not supported
```

桥接模式

桥接（Bridge）是用于把抽象化与实现化解耦，使得二者可以独立变化。这种类型的设计模式属于结构型模式，它通过提供抽象化和实现化之间的桥接结构，来实现二者的解耦。

这种模式涉及到一个作为桥接的接口，使得实体类的功能独立于接口实现类。这两种类型的类可被结构化改变而互不影响。

我们通过下面的实例来演示桥接模式（Bridge Pattern）的用法。其中，可以使用相同的抽象类方法但是不同的桥接实现类，来画出不同颜色的圆。

介绍

意图：将抽象部分与实现部分分离，使它们都可以独立的变化。

主要解决：在有多种可能会变化的情况下，用继承会造成类爆炸问题，扩展起来不灵活。

何时使用：实现系统可能有多个角度分类，每一种角度都可能变化。

如何解决：把这种多角度分类分离出来，让它们独立变化，减少它们之间耦合。

关键代码：抽象类依赖实现类。

应用实例： 1、猪八戒从天蓬元帅转世投胎到猪，转世投胎的机制将尘世划分为两个等级，即：灵魂和肉体，前者相当于抽象化，后者相当于实现化。生灵通过功能的委派，调用肉体对象的功能，使得生灵可以动态地选择。 2、墙上的开关，可以看到的开关是抽象的，不用管里面具体怎么实现的。

优点： 1、抽象和实现的分离。 2、优秀的扩展能力。 3、实现细节对客户透明。

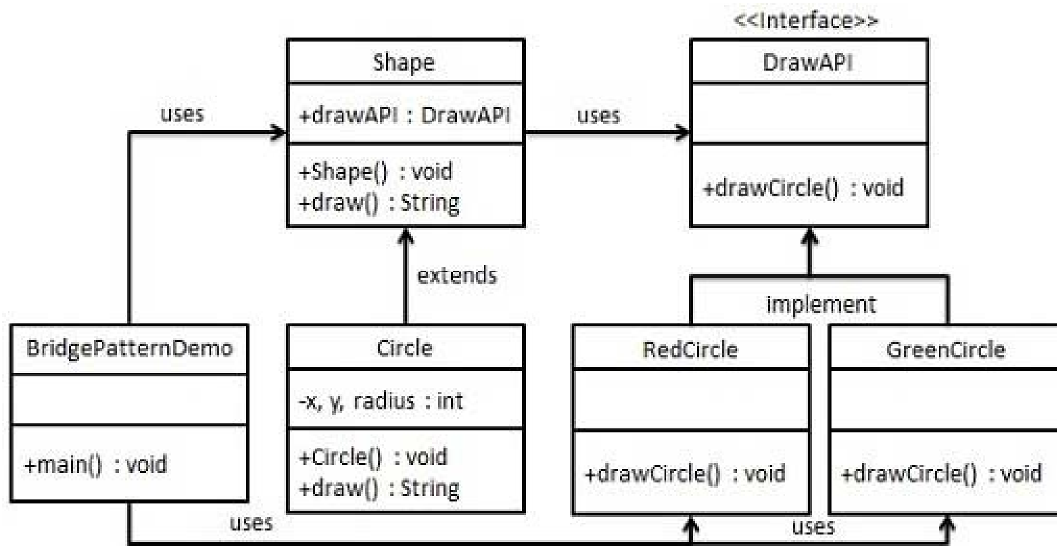
缺点：桥接模式的引入会增加系统的理解与设计难度，由于聚合关联关系建立在抽象层，要求开发者针对抽象进行设计与编程。

使用场景： 1、如果一个系统需要在构件的抽象化角色和具体化角色之间增加更多的灵活性，避免在两个层次之间建立静态的继承联系，通过桥接模式可以使它们在抽象层建立一个关联关系。 2、对于那些不希望使用继承或因为多层次继承导致系统类的个数急剧增加的系统，桥接模式尤为适用。 3、一个类存在两个独立变化的维度，且这两个维度都需要进行扩展。

注意事项：对于两个独立变化的维度，使用桥接模式再适合不过了。

实现

我们有一个作为桥接实现的 *DrawAPI* 接口和实现了 *DrawAPI* 接口的实体类 *RedCircle*、*GreenCircle*。*Shape* 是一个抽象类，将使用 *DrawAPI* 的对象。*BridgePatternDemo*，我们的演示类使用 *Shape* 类来画出不同颜色的圆。



步骤 1

创建桥接实现接口。

DrawAPI.java

```
public interface DrawAPI {
   public void drawCircle(int radius, int x, int y);
}
```

步骤 2

创建实现了 *DrawAPI* 接口的实体桥接实现类。

RedCircle.java

```
public class RedCircle implements DrawAPI {
   @Override
   public void drawCircle(int radius, int x, int y) {
      System.out.println("Drawing Circle[ color: red, radius: "
         + radius +", x: " +x+", "+ y +"]");
   }
}
```

GreenCircle.java

```
public class GreenCircle implements DrawAPI {
   @Override
   public void drawCircle(int radius, int x, int y) {
      System.out.println("Drawing Circle[ color: green, radius: "
         + radius +", x: " +x+", "+ y +"]");
   }
}
```

步骤 3

使用 *DrawAPI* 接口创建抽象类 *Shape*。

Shape.java

```
public abstract class Shape {
   protected DrawAPI drawAPI;
   protected Shape(DrawAPI drawAPI){
      this.drawAPI = drawAPI;
   }
   public abstract void draw();
}
```

步骤 4

创建实现了 *Shape* 接口的实体类。

Circle.java

```
public class Circle extends Shape {
   private int x, y, radius;

   public Circle(int x, int y, int radius, DrawAPI drawAPI) {
      super(drawAPI);
      this.x = x;
      this.y = y;
      this.radius = radius;
   }

   public void draw() {
      drawAPI.drawCircle(radius,x,y);
   }
}
```

步骤 5

使用 *Shape* 和 *DrawAPI* 类画出不同颜色的圆。

BridgePatternDemo.java

```
public class BridgePatternDemo {
   public static void main(String[] args) {
      Shape redCircle = new Circle(100,100, 10, new RedCircle());
      Shape greenCircle = new Circle(100,100, 10, new GreenCircle());

      redCircle.draw();
      greenCircle.draw();
   }
}
```

步骤 6

验证输出。

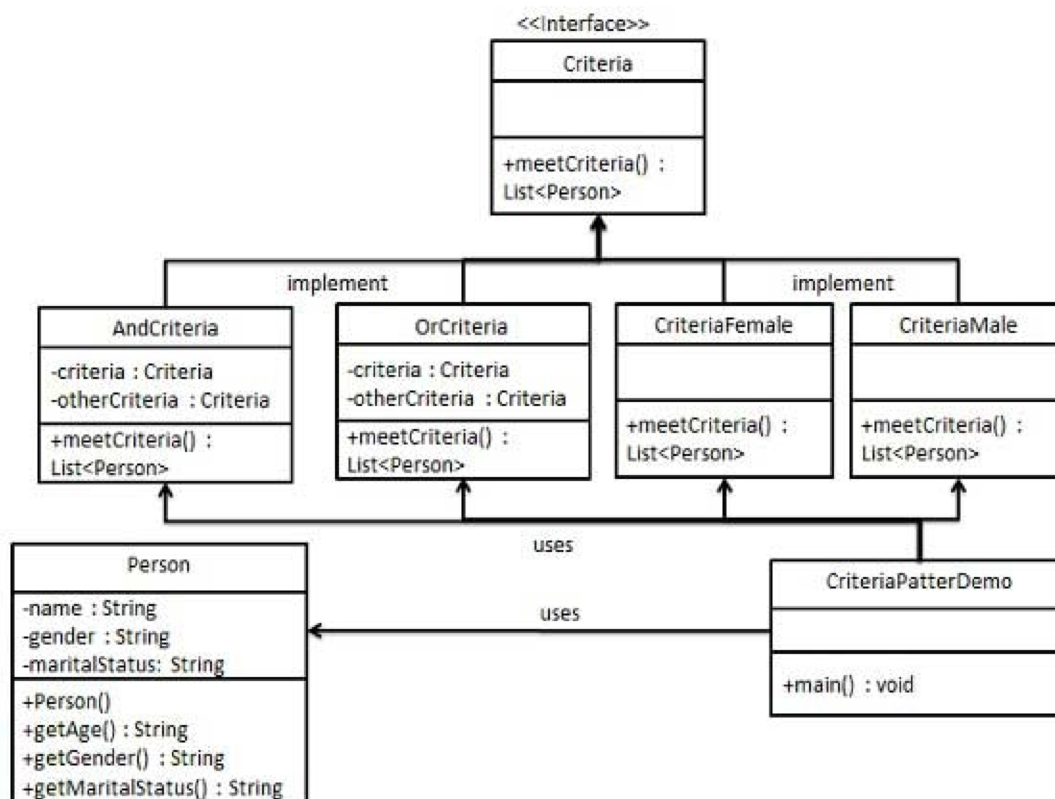
```
Drawing Circle[ color: red, radius: 10, x: 100, 100]
Drawing Circle[  color: green, radius: 10, x: 100, 100]
```

过滤器模式

过滤器模式（Filter Pattern）或标准模式（Criteria Pattern）是一种设计模式，这种模式允许开发人员使用不同的标准来过滤一组对象，通过逻辑运算以解耦的方式把它们连接起来。这种类型的设计模式属于结构型模式，它结合多个标准来获得单一标准。

实现

我们将创建一个 *Person* 对象、*Criteria* 接口和实现了该接口的实体类，来过滤 *Person* 对象的列表。*CriteriaPatternDemo*，我们的演示类使用 *Criteria* 对象，基于各种标准和它们的结合来过滤 *Person* 对象的列表。



步骤 1

创建一个类，在该类上应用标准。

Person.java

```
public class Person {

   private String name;
   private String gender;
   private String maritalStatus;

   public Person(String name,String gender,String maritalStatus){
      this.name = name;
      this.gender = gender;
      this.maritalStatus = maritalStatus;
   }

   public String getName() {
      return name;
   }
   public String getGender() {
      return gender;
   }
   public String getMaritalStatus() {
      return maritalStatus;
   }
}
```

步骤 2

为标准（Criteria）创建一个接口。

Criteria.java

```
import java.util.List;

public interface Criteria {
   public List<Person> meetCriteria(List<Person> persons);
}
```

步骤 3

创建实现了 *Criteria* 接口的实体类。

CriteriaMale.java

```
import java.util.ArrayList;
import java.util.List;

public class CriteriaMale implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> malePersons = new ArrayList<Person>();
      for (Person person : persons) {
         if(person.getGender().equalsIgnoreCase("MALE")){
            malePersons.add(person);
         }
      }
      return malePersons;
   }
}
```

CriteriaFemale.java

```
import java.util.ArrayList;
import java.util.List;

public class CriteriaFemale implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> femalePersons = new ArrayList<Person>();
      for (Person person : persons) {
         if(person.getGender().equalsIgnoreCase("FEMALE")){
            femalePersons.add(person);
         }
      }
      return femalePersons;
   }
}
```

CriteriaSingle.java

```
import java.util.ArrayList;
import java.util.List;

public class CriteriaSingle implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> singlePersons = new ArrayList<Person>();
      for (Person person : persons) {
         if(person.getMaritalStatus().equalsIgnoreCase("SINGLE")){
            singlePersons.add(person);
         }
      }
      return singlePersons;
   }
}
```

AndCriteria.java

```
import java.util.List;

public class AndCriteria implements Criteria {

   private Criteria criteria;
   private Criteria otherCriteria;

   public AndCriteria(Criteria criteria, Criteria otherCriteria) {
      this.criteria = criteria;
      this.otherCriteria = otherCriteria;
   }

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> firstCriteriaPersons = criteria.meetCriteria(persons);
      return otherCriteria.meetCriteria(firstCriteriaPersons);
   }
}
```

OrCriteria.java

```
import java.util.List;

public class OrCriteria implements Criteria {

   private Criteria criteria;
   private Criteria otherCriteria;

   public OrCriteria(Criteria criteria, Criteria otherCriteria) {
      this.criteria = criteria;
      this.otherCriteria = otherCriteria;
   }

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> firstCriteriaItems = criteria.meetCriteria(persons);
      List<Person> otherCriteriaItems = otherCriteria.meetCriteria(persons);

      for (Person person : otherCriteriaItems) {
         if(!firstCriteriaItems.contains(person)){
            firstCriteriaItems.add(person);
         }
      }
      return firstCriteriaItems;
   }
}
```

步骤4

使用不同的标准（Criteria）和它们的结合来过滤 *Person* 对象的列表。

CriteriaPatternDemo.java

```
public class CriteriaPatternDemo {
   public static void main(String[] args) {
      List<Person> persons = new ArrayList<Person>();

      persons.add(new Person("Robert","Male", "Single"));
      persons.add(new Person("John","Male", "Married"));
      persons.add(new Person("Laura","Female", "Married"));
      persons.add(new Person("Diana","Female", "Single"));
      persons.add(new Person("Mike","Male", "Single"));
      persons.add(new Person("Bobby","Male", "Single"));

      Criteria male = new CriteriaMale();
      Criteria female = new CriteriaFemale();
      Criteria single = new CriteriaSingle();
      Criteria singleMale = new AndCriteria(single, male);
      Criteria singleOrFemale = new OrCriteria(single, female);

      System.out.println("Males: ");
      printPersons(male.meetCriteria(persons));

      System.out.println("\nFemales: ");
      printPersons(female.meetCriteria(persons));

      System.out.println("\nSingle Males: ");
      printPersons(singleMale.meetCriteria(persons));

      System.out.println("\nSingle Or Females: ");
      printPersons(singleOrFemale.meetCriteria(persons));
   }

   public static void printPersons(List<Person> persons){
      for (Person person : persons) {
         System.out.println("Person : [ Name : " + person.getName()
            +", Gender : " + person.getGender()
            +", Marital Status : " + person.getMaritalStatus()
            +" ]");
      }
   }
}
```

步骤 5

验证输出。

```
Males:
Person : [ Name : Robert, Gender : Male, Marital Status : Single ]
Person : [ Name : John, Gender : Male, Marital Status : Married ]
Person : [ Name : Mike, Gender : Male, Marital Status : Single ]
Person : [ Name : Bobby, Gender : Male, Marital Status : Single ]

Females:
Person : [ Name : Laura, Gender : Female, Marital Status : Married ]
Person : [ Name : Diana, Gender : Female, Marital Status : Single ]

Single Males:
Person : [ Name : Robert, Gender : Male, Marital Status : Single ]
Person : [ Name : Mike, Gender : Male, Marital Status : Single ]
Person : [ Name : Bobby, Gender : Male, Marital Status : Single ]

Single Or Females:
Person : [ Name : Robert, Gender : Male, Marital Status : Single ]
Person : [ Name : Diana, Gender : Female, Marital Status : Single ]
Person : [ Name : Mike, Gender : Male, Marital Status : Single ]
Person : [ Name : Bobby, Gender : Male, Marital Status : Single ]
Person : [ Name : Laura, Gender : Female, Marital Status : Married ]
```

组合模式

组合模式（Composite Pattern），又叫部分整体模式，是用于把一组相似的对象当作一个单一的对象。组合模式依据树形结构来组合对象，用来表示部分以及整体层次。这种类型的设计模式属于结构型模式，它创建了对象组的树形结构。

这种模式创建了一个包含自己对象组的类。该类提供了修改相同对象组的方式。

我们通过下面的实例来演示组合模式的用法。实例演示了一个组织中员工的层次结构。

介绍

意图：将对象组合成树形结构以表示"部分-整体"的层次结构。组合模式使得用户对单个对象和组合对象的使用具有一致性。

主要解决：它在我们树型结构的问题中，模糊了简单元素和复杂元素的概念，客户程序可以向处理简单元素一样来处理复杂元素，从而使得客户程序与复杂元素的内部结构解耦。

何时使用： 1、您想表示对象的部分-整体层次结构（树形结构）。 2、您希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。

如何解决：树枝和叶子实现统一接口，树枝内部组合该接口。

关键代码：树枝内部组合该接口，并且含有内部属性 List，里面放 Component。

应用实例： 1、算术表达式包括操作数、操作符和另一个操作数，其中，另一个操作符也可以是操作树、操作符和另一个操作数。 2、在 JAVA AWT 和 SWING 中，对于 Button 和 Checkbox 是树叶，Container 是树枝。

优点： 1、高层模块调用简单。 2、节点自由增加。

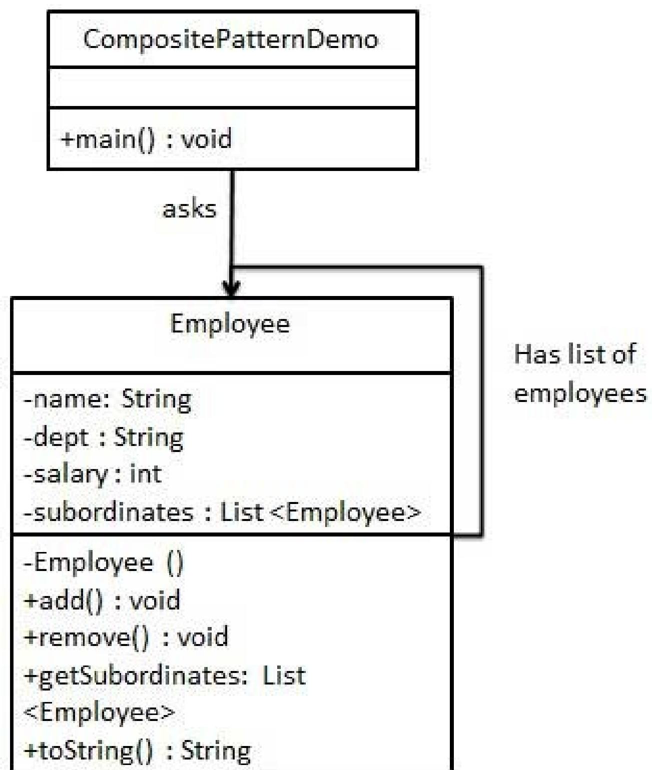
缺点：在使用组合模式时，其叶子和树枝的声明都是实现类，而不是接口，违反了依赖倒置原则。

使用场景：部分、整体场景，如树形菜单，文件、文件夹的管理。

注意事项：定义时为具体类。

实现

我们有一个类 *Employee*，该类被当作组合模型类。*CompositePatternDemo*，我们的演示类使用 *Employee* 类来添加部门层次结构，并打印所有员工。



步骤 1

创建 *Employee* 类，该类带有 *Employee* 对象的列表。

Employee.java

```
import java.util.ArrayList;
import java.util.List;

public class Employee {
   private String name;
   private String dept;
   private int salary;
   private List<Employee> subordinates;

   //构造函数
   public Employee(String name,String dept, int sal) {
      this.name = name;
      this.dept = dept;
      this.salary = sal;
      subordinates = new ArrayList<Employee>();
   }

   public void add(Employee e) {
      subordinates.add(e);
   }

   public void remove(Employee e) {
      subordinates.remove(e);
   }

   public List<Employee> getSubordinates(){
     return subordinates;
   }

   public String toString(){
      return ("Employee :[ Name : "+ name
      +", dept : "+ dept + ", salary :"
      + salary+" ]");
   }
}
```

步骤 2

使用 *Employee* 类来创建和打印员工的层次结构。

CompositePatternDemo.java

```
public class CompositePatternDemo {
   public static void main(String[] args) {
      Employee CEO = new Employee("John","CEO", 30000);

      Employee headSales = new Employee("Robert","Head Sales", 20000);

      Employee headMarketing = new Employee("Michel","Head Marketing", 20000);

      Employee clerk1 = new Employee("Laura","Marketing", 10000);
      Employee clerk2 = new Employee("Bob","Marketing", 10000);

      Employee salesExecutive1 = new Employee("Richard","Sales", 10000);
      Employee salesExecutive2 = new Employee("Rob","Sales", 10000);

      CEO.add(headSales);
      CEO.add(headMarketing);

      headSales.add(salesExecutive1);
      headSales.add(salesExecutive2);

      headMarketing.add(clerk1);
      headMarketing.add(clerk2);

      //打印该组织的所有员工
      System.out.println(CEO);
      for (Employee headEmployee : CEO.getSubordinates()) {
         System.out.println(headEmployee);
         for (Employee employee : headEmployee.getSubordinates()) {
            System.out.println(employee);
         }
      }
   }
}
```

步骤 3

验证输出。

```
Employee :[ Name : John, dept : CEO, salary :30000 ]
Employee :[ Name : Robert, dept : Head Sales, salary :20000 ]
Employee :[ Name : Richard, dept : Sales, salary :10000 ]
Employee :[ Name : Rob, dept : Sales, salary :10000 ]
Employee :[ Name : Michel, dept : Head Marketing, salary :20000 ]
Employee :[ Name : Laura, dept : Marketing, salary :10000 ]
Employee :[ Name : Bob, dept : Marketing, salary :10000 ]
```

装饰器模式

装饰器模式（Decorator Pattern）允许向一个现有的对象添加新的功能，同时又不改变其结构。这种类型的设计模式属于结构型模式，它是作为现有的类的一个包装。

这种模式创建了一个装饰类，用来包装原有的类，并在保持类方法签名完整性的前提下，提供了额外的功能。

我们通过下面的实例来演示装饰器模式的用法。其中，我们将把一个形状装饰上不同的颜色，同时又不改变形状类。

介绍

意图：动态地给一个对象添加一些额外的职责。就增加功能来说，装饰器模式相比生成子类更为灵活。

主要解决：一般的，我们为了扩展一个类经常使用继承方式实现，由于继承为类引入静态特征，并且随着扩展功能的增多，子类会很膨胀。

何时使用：在不想增加很多子类的情况下扩展类。

如何解决：将具体功能职责划分，同时继承装饰者模式。

关键代码： 1、Component 类充当抽象角色，不应该具体实现。 2、修饰类引用和继承 Component 类，具体扩展类重写父类方法。

应用实例： 1、孙悟空有 72 变，当他变成"庙宇"后，他的根本还是一只猴子，但是他又有了庙宇的功能。 2、不论一幅画有没有画框都可以挂在墙上，但是通常都是有画框的，并且实际上是画框被挂在墙上。在挂在墙上之前，画可以被蒙上玻璃，装到框子里；这时画、玻璃和画框形成了一个物体。

优点：装饰类和被装饰类可以独立发展，不会相互耦合，装饰模式是继承的一个替代模式，装饰模式可以动态扩展一个实现类的功能。

缺点：多层装饰比较复杂。

使用场景： 1、扩展一个类的功能。 2、动态增加功能，动态撤销。

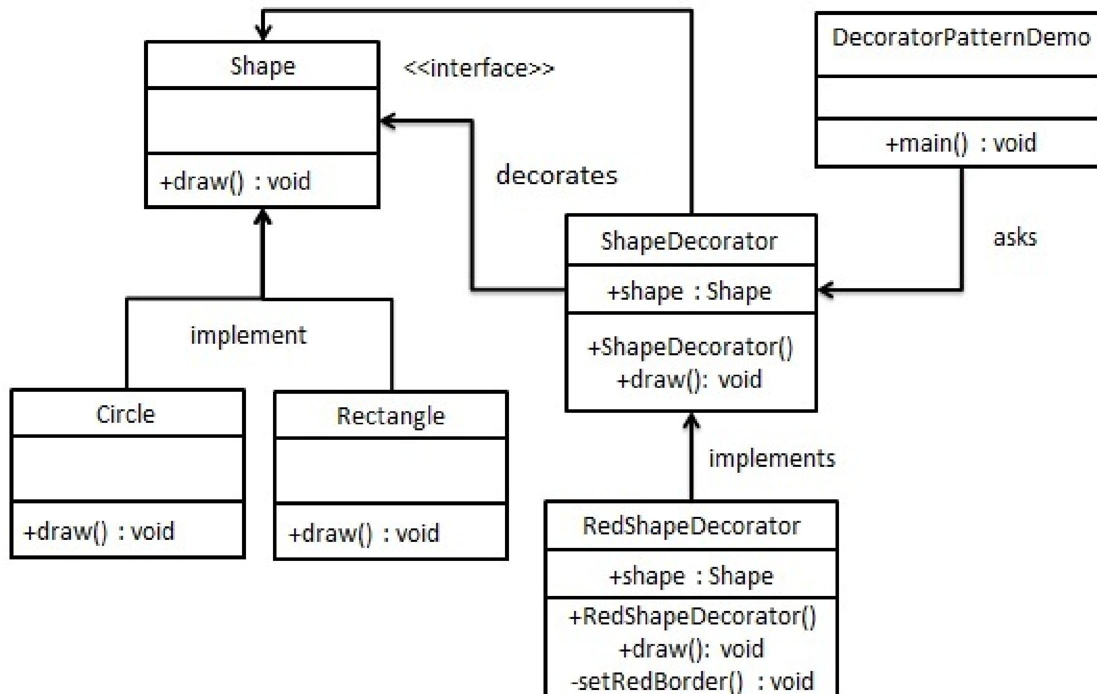
注意事项：可代替继承。

实现

我们将创建一个 *Shape* 接口和实现了 *Shape* 接口的实体类。然后我们创建一个实现了 *Shape* 接口的抽象装饰类 *ShapeDecorator*，并把 *Shape* 对象作为它的实例变量。

RedShapeDecorator 是实现了 *ShapeDecorator* 的实体类。

DecoratorPatternDemo，我们的演示类使用 *RedShapeDecorator* 来装饰 *Shape* 对象。



步骤 1

创建一个接口。

Shape.java

```
public interface Shape {
   void draw();
}
```

步骤 2

创建实现接口的实体类。

Rectangle.java

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Shape: Rectangle");
   }
}
```

Circle.java

```
public class Circle implements Shape {

   @Override
   public void draw() {
      System.out.println("Shape: Circle");
   }
}
```

步骤 3

创建实现了 *Shape* 接口的抽象装饰类。

ShapeDecorator.java

```
public abstract class ShapeDecorator implements Shape {
   protected Shape decoratedShape;

   public ShapeDecorator(Shape decoratedShape){
      this.decoratedShape = decoratedShape;
   }

   public void draw(){
      decoratedShape.draw();
   }
}
```

步骤 4

创建扩展了 *ShapeDecorator* 类的实体装饰类。

RedShapeDecorator.java

```
public class RedShapeDecorator extends ShapeDecorator {

   public RedShapeDecorator(Shape decoratedShape) {
      super(decoratedShape);
   }

   @Override
   public void draw() {
      decoratedShape.draw();
      setRedBorder(decoratedShape);
   }

   private void setRedBorder(Shape decoratedShape){
      System.out.println("Border Color: Red");
   }
}
```

步骤 5

使用 *RedShapeDecorator* 来装饰 *Shape* 对象。

DecoratorPatternDemo.java

```
public class DecoratorPatternDemo {
   public static void main(String[] args) {

      Shape circle = new Circle();

      Shape redCircle = new RedShapeDecorator(new Circle());

      Shape redRectangle = new RedShapeDecorator(new Rectangle());
      System.out.println("Circle with normal border");
      circle.draw();

      System.out.println("\nCircle of red border");
      redCircle.draw();

      System.out.println("\nRectangle of red border");
      redRectangle.draw();
   }
}
```

步骤 6

验证输出。

```
Circle with normal border
Shape: Circle

Circle of red border
Shape: Circle
Border Color: Red

Rectangle of red border
Shape: Rectangle
Border Color: Red
```

外观模式

外观模式（Facade Pattern）隐藏系统的复杂性，并向客户端提供了一个客户端可以访问系统的接口。这种类型的设计模式属于结构型模式，它向现有的系统添加一个接口，来隐藏系统的复杂性。

这种模式涉及到一个单一的类，该类提供了客户端请求的简化方法和对现有系统类方法的委托调用。

介绍

意图：为子系统中的一组接口提供一个一致的界面，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。

主要解决：降低访问复杂系统的内部子系统时的复杂度，简化客户端与之的接口。

何时使用： 1、客户端不需要知道系统内部的复杂联系，整个系统只需提供一个"接待员"即可。 2、定义系统的入口。

如何解决：客户端不与系统耦合，外观类与系统耦合。

关键代码：在客户端和复杂系统之间再加一层，这一次将调用顺序、依赖关系等处理好。

应用实例： 1、去医院看病，可能要去挂号、门诊、划价、取药，让患者或患者家属觉得很复杂，如果有提供接待人员，只让接待人员来处理，就很方便。 2、JAVA 的三层开发模式。

优点： 1、减少系统相互依赖。 2、提高灵活性。 3、提高了安全性。

缺点：不符合开闭原则，如果要改东西很麻烦，继承重写都不合适。

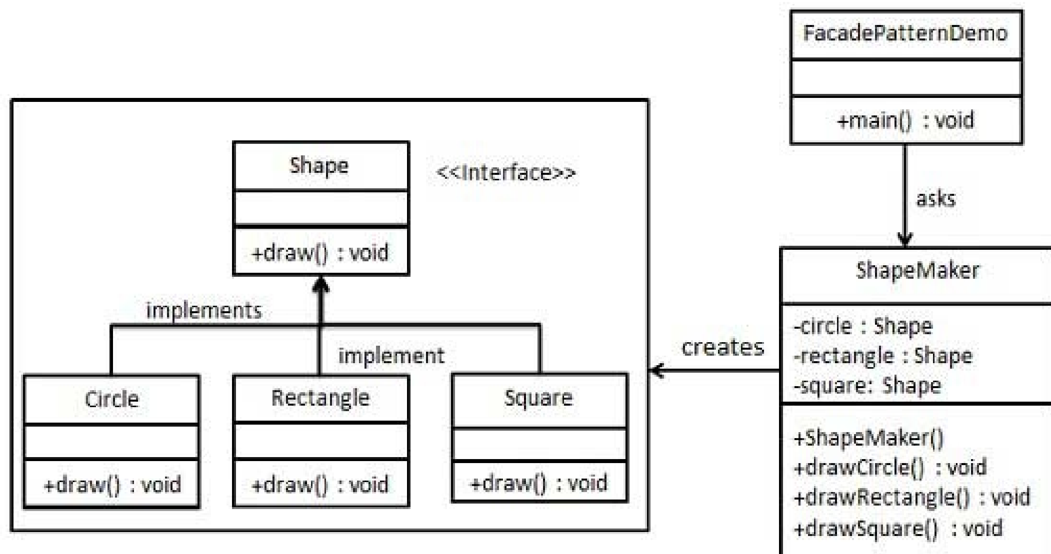
使用场景： 1、为复杂的模块或子系统提供外界访问的模块。 2、子系统相对独立。 3、预防低水平人员带来的风险。

注意事项：在层次化结构中，可以使用外观模式定义系统中每一层的入口。

实现

我们将创建一个 *Shape* 接口和实现了 *Shape* 接口的实体类。下一步是定义一个外观类 *ShapeMaker*。

ShapeMaker 类使用实体类来代表用户对这些类的调用。*FacadePatternDemo*，我们的演示类使用 *ShapeMaker* 类来显示结果。



步骤 1

创建一个接口。

Shape.java

```
public interface Shape {
   void draw();
}
```

步骤 2

创建实现接口的实体类。

Rectangle.java

```
public class Rectangle implements Shape {

   @Override
   public void draw() {
      System.out.println("Rectangle::draw()");
   }
}
```

Square.java

```
public class Square implements Shape {

   @Override
   public void draw() {
      System.out.println("Square::draw()");
   }
}
```

Circle.java

```
public class Circle implements Shape {

   @Override
   public void draw() {
      System.out.println("Circle::draw()");
   }
}
```

步骤 3

创建一个外观类。

ShapeMaker.java

```
public class ShapeMaker {
   private Shape circle;
   private Shape rectangle;
   private Shape square;

   public ShapeMaker() {
      circle = new Circle();
      rectangle = new Rectangle();
      square = new Square();
   }

   public void drawCircle(){
      circle.draw();
   }
   public void drawRectangle(){
      rectangle.draw();
   }
   public void drawSquare(){
      square.draw();
   }
}
```

步骤 4

使用该外观类画出各种类型的形状。

FacadePatternDemo.java

```
public class FacadePatternDemo {
   public static void main(String[] args) {
      ShapeMaker shapeMaker = new ShapeMaker();

      shapeMaker.drawCircle();
      shapeMaker.drawRectangle();
      shapeMaker.drawSquare();
   }
}
```

步骤 5

验证输出。

```
Circle::draw()
Rectangle::draw()
Square::draw()
```

享元模式

享元模式（Flyweight Pattern）主要用于减少创建对象的数量，以减少内存占用和提高性能。这种类型的设计模式属于结构型模式，它提供了减少对象数量从而改善应用所需的对象结构的方式。

享元模式尝试重用现有的同类对象，如果未找到匹配的对象，则创建新对象。我们将通过创建 5 个对象来画出 20 个分布于不同位置的圆来演示这种模式。由于只有 5 种可用的颜色，所以 color 属性被用来检查现有的 *Circle* 对象。

介绍

意图：运用共享技术有效地支持大量细粒度的对象。

主要解决：在有大量对象时，有可能会造成内存溢出，我们把其中共同的部分抽象出来，如果有相同的业务请求，直接返回在内存中已有的对象，避免重新创建。

何时使用： 1、系统中有大量对象。 2、这些对象消耗大量内存。 3、这些对象的状态大部分可以外部化。 4、这些对象可以按照内蕴状态分为很多组，当把外蕴对象从对象中剔除出来时，每一组对象都可以用一个对象来代替。 5、系统不依赖于这些对象身份，这些对象是不可分辨的。

如何解决：用唯一标识码判断，如果在内存中有，则返回这个唯一标识码所标识的对象。

关键代码：用 HashMap 存储这些对象。

应用实例： 1、JAVA 中的 String，如果有则返回，如果没有则创建一个字符串保存在字符串缓存池里面。 2、数据库的数据池。

优点：大大减少对象的创建，降低系统的内存，使效率提高。

缺点：提高了系统的负责度，需要分离出外部状态和内部状态，而且外部状态具有固有化的性质，不应该随着内部状态的变化而变化，否则会造成系统的混乱。

使用场景： 1、系统有大量相似对象。 2、需要缓冲池的场景。

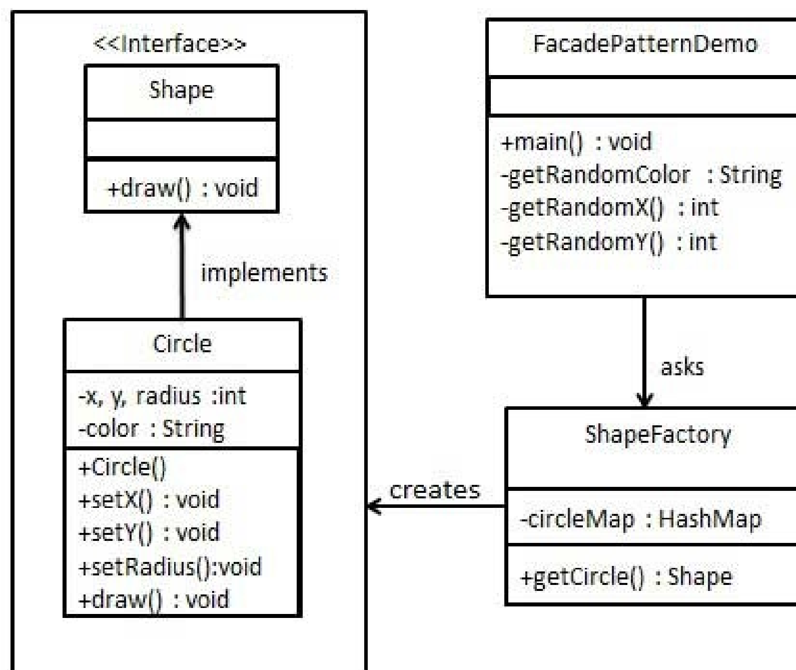
注意事项： 1、注意划分外部状态和内部状态，否则可能会引起线程安全问题。 2、这些类必须有一个工厂对象加以控制。

实现

我们将创建一个 *Shape* 接口和实现了 *Shape* 接口的实体类 *Circle*。下一步是定义工厂类 *ShapeFactory*。

ShapeFactory 有一个 *Circle* 的 *HashMap*，其中键名为 *Circle* 对象的颜色。无论何时接收到请求，都会创建一个特定颜色的圆。*ShapeFactory* 检查它的 *HashMap* 中的 circle 对象，如果找到 *Circle* 对象，则返回该对象，否则将创建一个存储在 hashmap 中以备后续使用的新对象，并把该对象返回到客户端。

FlyWeightPatternDemo，我们的演示类使用 *ShapeFactory* 来获取 *Shape* 对象。它将向 *ShapeFactory* 传递信息（*red / green / blue/ black / white*），以便获取它所需对象的颜色。



步骤 1

创建一个接口。

Shape.java

```
public interface Shape {
   void draw();
}
```

步骤 2

创建实现接口的实体类。

Circle.java

```
public class Circle implements Shape {
   private String color;
   private int x;
   private int y;
   private int radius;

   public Circle(String color){
      this.color = color;
   }

   public void setX(int x) {
      this.x = x;
   }

   public void setY(int y) {
      this.y = y;
   }

   public void setRadius(int radius) {
      this.radius = radius;
   }

   @Override
   public void draw() {
      System.out.println("Circle: Draw() [Color : " + color
         +", x : " + x +", y :" + y +", radius :" + radius);
   }
}
```

步骤 3

创建一个工厂，生成基于给定信息的实体类的对象。

ShapeFactory.java

```
import java.util.HashMap;

public class ShapeFactory {
   private static final HashMap<String, Shape> circleMap = new HashMap();

   public static Shape getCircle(String color) {
      Circle circle = (Circle)circleMap.get(color);

      if(circle == null) {
         circle = new Circle(color);
         circleMap.put(color, circle);
         System.out.println("Creating circle of color : " + color);
      }
      return circle;
   }
}
```

步骤 4

使用该工厂，通过传递颜色信息来获取实体类的对象。

FlyweightPatternDemo.java

```
public class FlyweightPatternDemo {
   private static final String colors[] =
      { "Red", "Green", "Blue", "White", "Black" };
   public static void main(String[] args) {

      for(int i=0; i < 20; ++i) {
         Circle circle =
            (Circle)ShapeFactory.getCircle(getRandomColor());
         circle.setX(getRandomX());
         circle.setY(getRandomY());
         circle.setRadius(100);
         circle.draw();
      }
   }
   private static String getRandomColor() {
      return colors[(int)(Math.random()*colors.length)];
   }
   private static int getRandomX() {
      return (int)(Math.random()*100 );
   }
   private static int getRandomY() {
      return (int)(Math.random()*100);
   }
}
```

步骤 5

验证输出。

```
Creating circle of color : Black
Circle: Draw() [Color : Black, x : 36, y :71, radius :100
Creating circle of color : Green
Circle: Draw() [Color : Green, x : 27, y :27, radius :100
Creating circle of color : White
Circle: Draw() [Color : White, x : 64, y :10, radius :100
Creating circle of color : Red
Circle: Draw() [Color : Red, x : 15, y :44, radius :100
Circle: Draw() [Color : Green, x : 19, y :10, radius :100
Circle: Draw() [Color : Green, x : 94, y :32, radius :100
Circle: Draw() [Color : White, x : 69, y :98, radius :100
Creating circle of color : Blue
Circle: Draw() [Color : Blue, x : 13, y :4, radius :100
Circle: Draw() [Color : Green, x : 21, y :21, radius :100
Circle: Draw() [Color : Blue, x : 55, y :86, radius :100
Circle: Draw() [Color : White, x : 90, y :70, radius :100
Circle: Draw() [Color : Green, x : 78, y :3, radius :100
Circle: Draw() [Color : Green, x : 64, y :89, radius :100
Circle: Draw() [Color : Blue, x : 3, y :91, radius :100
Circle: Draw() [Color : Blue, x : 62, y :82, radius :100
Circle: Draw() [Color : Green, x : 97, y :61, radius :100
Circle: Draw() [Color : Green, x : 86, y :12, radius :100
Circle: Draw() [Color : Green, x : 38, y :93, radius :100
Circle: Draw() [Color : Red, x : 76, y :82, radius :100
Circle: Draw() [Color : Blue, x : 95, y :82, radius :100
```

代理模式

在代理模式（Proxy Pattern）中，一个类代表另一个类的功能。这种类型的设计模式属于结构型模式。

在代理模式中，我们创建具有现有对象的对象，以便向外界提供功能接口。

介绍

意图：为其他对象提供一种代理以控制对这个对象的访问。

主要解决：在直接访问对象时带来的问题，比如说：要访问的对象在远程的机器上。在面向对象系统中，有些对象由于某些原因（比如对象创建开销很大，或者某些操作需要安全控制，或者需要进程外的访问），直接访问会给使用者或者系统结构带来很多麻烦，我们可以在访问此对象时加上一个对此对象的访问层。

何时使用：想在访问一个类时做一些控制。

如何解决：增加中间层。

关键代码：实现与被代理类组合。

应用实例： 1、Windows 里面的快捷方式。 2、猪八戒去找高翠兰结果是孙悟空变的，可以这样理解：把高翠兰的外貌抽象出来，高翠兰本人和孙悟空都实现了这个接口，猪八戒访问高翠兰的时候看不出来这个是孙悟空，所以说孙悟空是高翠兰代理类。 3、买火车票不一定在火车站买，也可以去代售点。 4、一张支票或银行存单是账户中资金的代理。支票在市场交易中用来代替现金，并提供对签发人账号上资金的控制。 5、spring aop。

优点： 1、职责清晰。 2、高扩展性。 3、智能化。

缺点： 1、由于在客户端和真实主题之间增加了代理对象，因此有些类型的代理模式可能会造成请求的处理速度变慢。 2、实现代理模式需要额外的工作，有些代理模式的实现非常复杂。

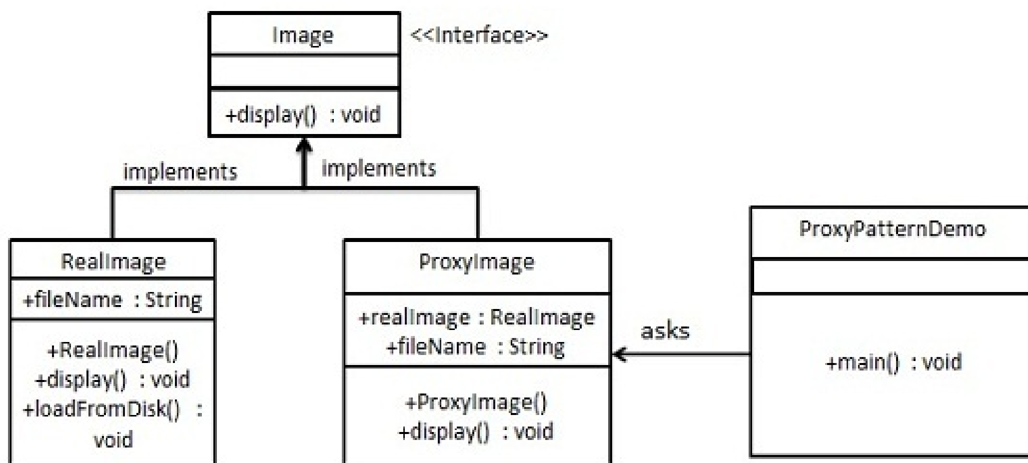
使用场景：按职责来划分，通常有以下使用场景： 1、远程代理。 2、虚拟代理。 3、Copy-on-Write 代理。 4、保护（Protect or Access）代理。 5、Cache代理。 6、防火墙（Firewall）代理。 7、同步化（Synchronization）代理。 8、智能引用（Smart Reference）代理。

注意事项： 1、和适配器模式的区别：适配器模式主要改变所考虑对象的接口，而代理模式不能改变所代理类的接口。 2、和装饰器模式的区别：装饰器模式为了增强功能，而代理模式是为了加以控制。

实现

我们将创建一个 *Image* 接口和实现了 *Image* 接口的实体类。*ProxyImage* 是一个代理类，减少 *RealImage* 对象加载的内存占用。

ProxyPatternDemo，我们的演示类使用 *ProxyImage* 来获取要加载的 *Image* 对象，并按照需求进行显示。



步骤 1

创建一个接口。

Image.java

```
public interface Image {
   void display();
}
```

步骤 2

创建实现接口的实体类。

RealImage.java

```
public class RealImage implements Image {

   private String fileName;

   public RealImage(String fileName){
      this.fileName = fileName;
      loadFromDisk(fileName);
   }

   @Override
   public void display() {
      System.out.println("Displaying " + fileName);
   }

   private void loadFromDisk(String fileName){
      System.out.println("Loading " + fileName);
   }
}
```

ProxyImage.java

```
public class ProxyImage implements Image{

   private RealImage realImage;
   private String fileName;

   public ProxyImage(String fileName){
      this.fileName = fileName;
   }

   @Override
   public void display() {
      if(realImage == null){
         realImage = new RealImage(fileName);
      }
      realImage.display();
   }
}
```

步骤 3

当被请求时，使用 *ProxyImage* 来获取 *RealImage* 类的对象。

ProxyPatternDemo.java

```
public class ProxyPatternDemo {

   public static void main(String[] args) {
      Image image = new ProxyImage("test_10mb.jpg");

      //图像将从磁盘加载
      image.display();
      System.out.println("");
      //图像将无法从磁盘加载
      image.display();
   }
}
```

步骤 4

验证输出。

```
Loading test_10mb.jpg
Displaying test_10mb.jpg

Displaying test_10mb.jpg
```

责任链模式

顾名思义，责任链模式（Chain of Responsibility Pattern）为请求创建了一个接收者对象的链。这种模式给予请求的类型，对请求的发送者和接收者进行解耦。这种类型的设计模式属于行为型模式。

在这种模式中，通常每个接收者都包含对另一个接收者的引用。如果一个对象不能处理该请求，那么它会把相同的请求传给下一个接收者，依此类推。

介绍

意图：避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。

主要解决：职责链上的处理者负责处理请求，客户只需要将请求发送到职责链上即可，无须关心请求的处理细节和请求的传递，所以职责链将请求的发送者和请求的处理者解耦了。

何时使用：在处理消息的时候以过滤很多道。

如何解决：拦截的类都实现统一接口。

关键代码：Handler 里面聚合它自己，在 HanleRequest 里判断是否合适，如果没达到条件则向下传递，向谁传递之前 set 进去。

应用实例： 1、红楼梦中的"击鼓传花"。 2、JS 中的事件冒泡。 3、JAVA WEB 中 Apache Tomcat 对 Encoding 的处理，Struts2 的拦截器，jsp servlet 的 Filter。

优点： 1、降低耦合度。它将请求的发送者和接收者解耦。 2、简化了对象。使得对象不需要知道链的结构。 3、增强给对象指派职责的灵活性。通过改变链内的成员或者调动它们的次序，允许动态地新增或者删除责任。 4、增加新的请求处理类很方便。

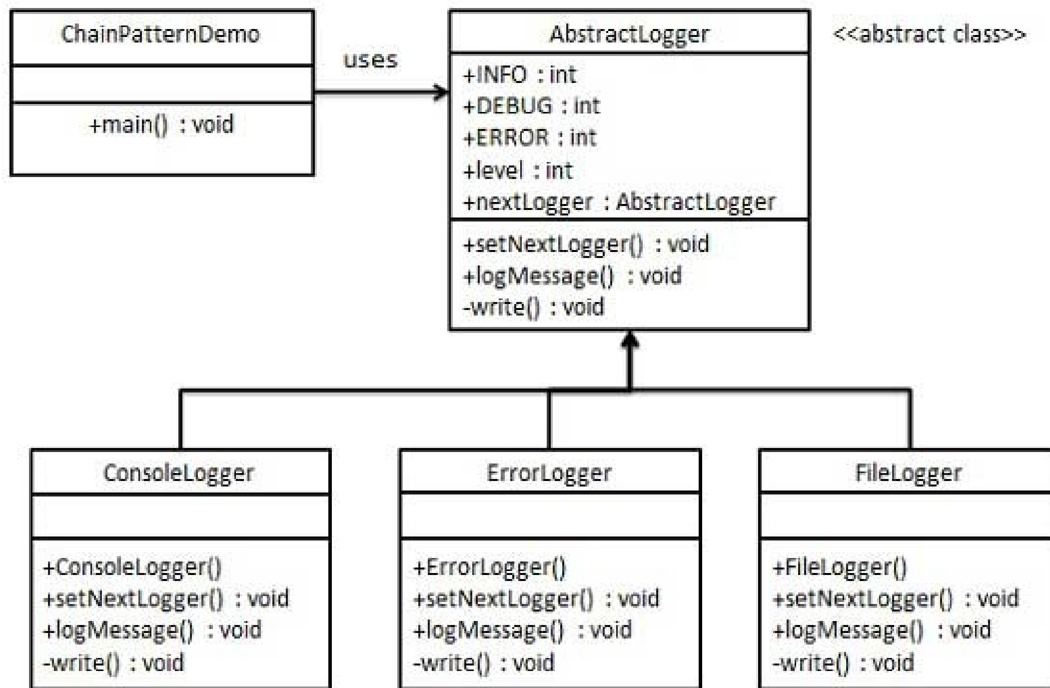
缺点： 1、不能保证请求一定被接收。 2、系统性能将受到一定影响，而且在进行代码调试时不太方便，可能会造成循环调用。 3、可能不容易观察运行时的特征，有碍于除错。

使用场景： 1、有多个对象可以处理同一个请求，具体哪个对象处理该请求由运行时刻自动确定。 2、在不明确指定接收者的情况下，向多个对象中的一个提交一个请求。 3、可动态指定一组对象处理请求。

注意事项：在 JAVA WEB 中遇到很多应用。

实现

我们创建抽象类 *AbstractLogger*，带有详细的日志记录级别。然后我们创建三种类型的记录器，都扩展了 *AbstractLogger*。每个记录器消息的级别是否属于自己的级别，如果是则相应地打印出来，否则将不打印并把消息传给下一个记录器。



步骤 1

创建抽象的记录器类。

AbstractLogger.java

```
public abstract class AbstractLogger {
   public static int INFO = 1;
   public static int DEBUG = 2;
   public static int ERROR = 3;

   protected int level;

   //责任链中的下一个元素
   protected AbstractLogger nextLogger;

   public void setNextLogger(AbstractLogger nextLogger){
      this.nextLogger = nextLogger;
   }

   public void logMessage(int level, String message){
      if(this.level <= level){
         write(message);
      }
      if(nextLogger !=null){
         nextLogger.logMessage(level, message);
      }
   }

   abstract protected void write(String message);

}
```

步骤 2

创建扩展了该记录器类的实体类。

ConsoleLogger.java

```
public class ConsoleLogger extends AbstractLogger {

   public ConsoleLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("Standard Console::Logger: " + message);
   }
}
```

ErrorLogger.java

```
public class ErrorLogger extends AbstractLogger {

   public ErrorLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("Error Console::Logger: " + message);
   }
}
```

FileLogger.java

```
public class FileLogger extends AbstractLogger {

   public FileLogger(int level){
      this.level = level;
   }

   @Override
   protected void write(String message) {
      System.out.println("File::Logger: " + message);
   }
}
```

步骤 3

创建不同类型的记录器。赋予它们不同的错误级别，并在每个记录器中设置下一个记录器。每个记录器中的下一个记录器代表的是链的一部分。

ChainPatternDemo.java

```
public class ChainPatternDemo {

   private static AbstractLogger getChainOfLoggers(){

      AbstractLogger errorLogger = new ErrorLogger(AbstractLogger.ERROR);
      AbstractLogger fileLogger = new FileLogger(AbstractLogger.DEBUG);
      AbstractLogger consoleLogger = new ConsoleLogger(AbstractLogger.INFO);

      errorLogger.setNextLogger(fileLogger);
      fileLogger.setNextLogger(consoleLogger);

      return errorLogger;
   }

   public static void main(String[] args) {
      AbstractLogger loggerChain = getChainOfLoggers();

      loggerChain.logMessage(AbstractLogger.INFO,
         "This is an information.");

      loggerChain.logMessage(AbstractLogger.DEBUG,
         "This is an debug level information.");

      loggerChain.logMessage(AbstractLogger.ERROR,
         "This is an error information.");
   }
}
```

步骤 4

验证输出。

```
Standard Console::Logger: This is an information.
File::Logger: This is an debug level information.
Standard Console::Logger: This is an debug level information.
Error Console::Logger: This is an error information.
File::Logger: This is an error information.
Standard Console::Logger: This is an error information.
```

命令模式

命令模式（Command Pattern）是一种数据驱动的设计模式，它属于行为型模式。请求以命令的形式包裹在对象中，并传给调用对象。调用对象寻找可以处理该命令的合适的对象，并把该命令传给相应的对象，该对象执行命令。

介绍

意图：将一个请求封装成一个对象，从而使您可以用不同的请求对客户进行参数化。

主要解决：在软件系统中，行为请求者与行为实现者通常是一种紧耦合的关系，但某些场合，比如需要对行为进行记录、撤销或重做、事务等处理时，这种无法抵御变化的紧耦合的设计就不太合适。

何时使用：在某些场合，比如要对行为进行"记录、撤销/重做、事务"等处理，这种无法抵御变化的紧耦合是不合适的。在这种情况下，如何将"行为请求者"与"行为实现者"解耦？将一组行为抽象为对象，可以实现二者之间的松耦合。

如何解决：通过调用者调用接受者执行命令，顺序：调用者→接受者→命令。

关键代码：定义三个角色：1、received 真正的命令执行对象 2、Command 3、invoker 使用命令对象的入口

应用实例：struts 1 中的 action 核心控制器 ActionServlet 只有一个，相当于 Invoker，而模型层的类会随着不同的应用有不同的模型类，相当于具体的 Command。

优点： 1、降低了系统耦合度。 2、新的命令可以很容易添加到系统中去。

缺点：使用命令模式可能会导致某些系统有过多的具体命令类。

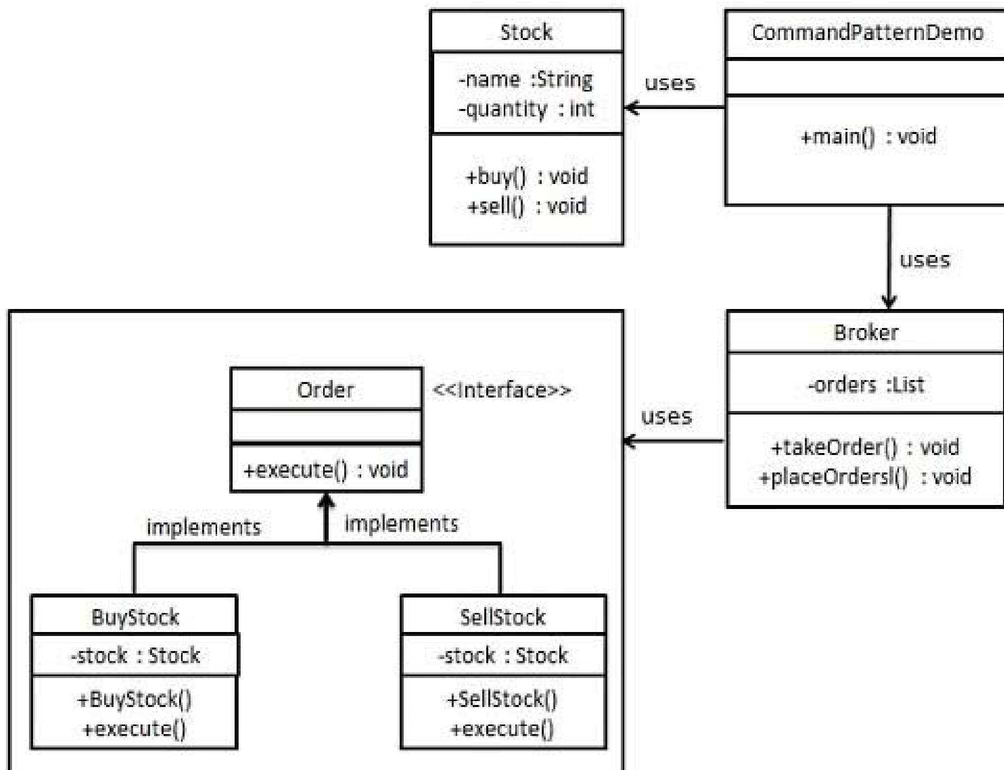
使用场景：认为是命令的地方都可以使用命令模式，比如： 1、GUI 中每一个按钮都是一条命令。 2、模拟 CMD。

注意事项：系统需要支持命令的撤销(Undo)操作和恢复(Redo)操作，也可以考虑使用命令模式，见命令模式的扩展。

实现

我们首先创建作为命令的接口 *Order*，然后创建作为请求的 *Stock* 类。实体命令类 *BuyStock* 和 *SellStock*，实现了 *Order* 接口，将执行实际的命令处理。创建作为调用对象的类 *Broker*，它接受订单并能下订单。

Broker 对象使用命令模式，基于命令的类型确定哪个对象执行哪个命令。*CommandPatternDemo*，我们的演示类使用 *Broker* 类来演示命令模式。



步骤 1

创建一个命令接口。

Order.java

```
public interface Order {
   void execute();
}
```

步骤 2

创建一个请求类。

Stock.java

```
public class Stock {

   private String name = "ABC";
   private int quantity = 10;

   public void buy(){
      System.out.println("Stock [ Name: "+name+",
         Quantity: " + quantity +" ] bought");
   }
   public void sell(){
      System.out.println("Stock [ Name: "+name+",
         Quantity: " + quantity +" ] sold");
   }
}
```

步骤 3

创建实现了 *Order* 接口的实体类。

BuyStock.java

```
public class BuyStock implements Order {
   private Stock abcStock;

   public BuyStock(Stock abcStock){
      this.abcStock = abcStock;
   }

   public void execute() {
      abcStock.buy();
   }
}
```

SellStock.java

```
public class SellStock implements Order {
   private Stock abcStock;

   public SellStock(Stock abcStock){
      this.abcStock = abcStock;
   }

   public void execute() {
      abcStock.sell();
   }
}
```

步骤 4

创建命令调用类。

Broker.java

```
import java.util.ArrayList;
import java.util.List;

   public class Broker {
   private List<Order> orderList = new ArrayList<Order>();

   public void takeOrder(Order order){
      orderList.add(order);
   }

   public void placeOrders(){
      for (Order order : orderList) {
         order.execute();
      }
      orderList.clear();
   }
}
```

步骤 5

使用 Broker 类来接受并执行命令。

CommandPatternDemo.java

```
public class CommandPatternDemo {
   public static void main(String[] args) {
      Stock abcStock = new Stock();

      BuyStock buyStockOrder = new BuyStock(abcStock);
      SellStock sellStockOrder = new SellStock(abcStock);

      Broker broker = new Broker();
      broker.takeOrder(buyStockOrder);
      broker.takeOrder(sellStockOrder);

      broker.placeOrders();
   }
}
```

步骤 6

验证输出。

```
Stock [ Name: ABC, Quantity: 10 ] bought
Stock [ Name: ABC, Quantity: 10 ] sold
```

解释器模式

解释器模式（Interpreter Pattern）提供了评估语言的语法或表达式的方式，它属于行为型模式。这种模式实现了一个表达式接口，该接口解释一个特定的上下文。这种模式被用在 SQL 解析、符号处理引擎等。

介绍

意图：给定一个语言，定义它的文法表示，并定义一个解释器，这个解释器使用该标识来解释语言中的句子。

主要解决：对于一些固定文法构建一个解释句子的解释器。

何时使用：如果一种特定类型的问题发生的频率足够高，那么可能就值得将该问题的各个实例表述为一个简单语言中的句子。这样就可以构建一个解释器，该解释器通过解释这些句子来解决该问题。

如何解决：构件语法树，定义终结符与非终结符。

关键代码：构件环境类，包含解释器之外的一些全局信息，一般是 HashMap。

应用实例：编译器、运算表达式计算。

优点： 1、可扩展性比较好，灵活。 2、增加了新的解释表达式的方式。 3、易于实现简单文法。

缺点： 1、可利用场景比较少。 2、对于复杂的文法比较难维护。 3、解释器模式会引起类膨胀。 4、解释器模式采用递归调用方法。

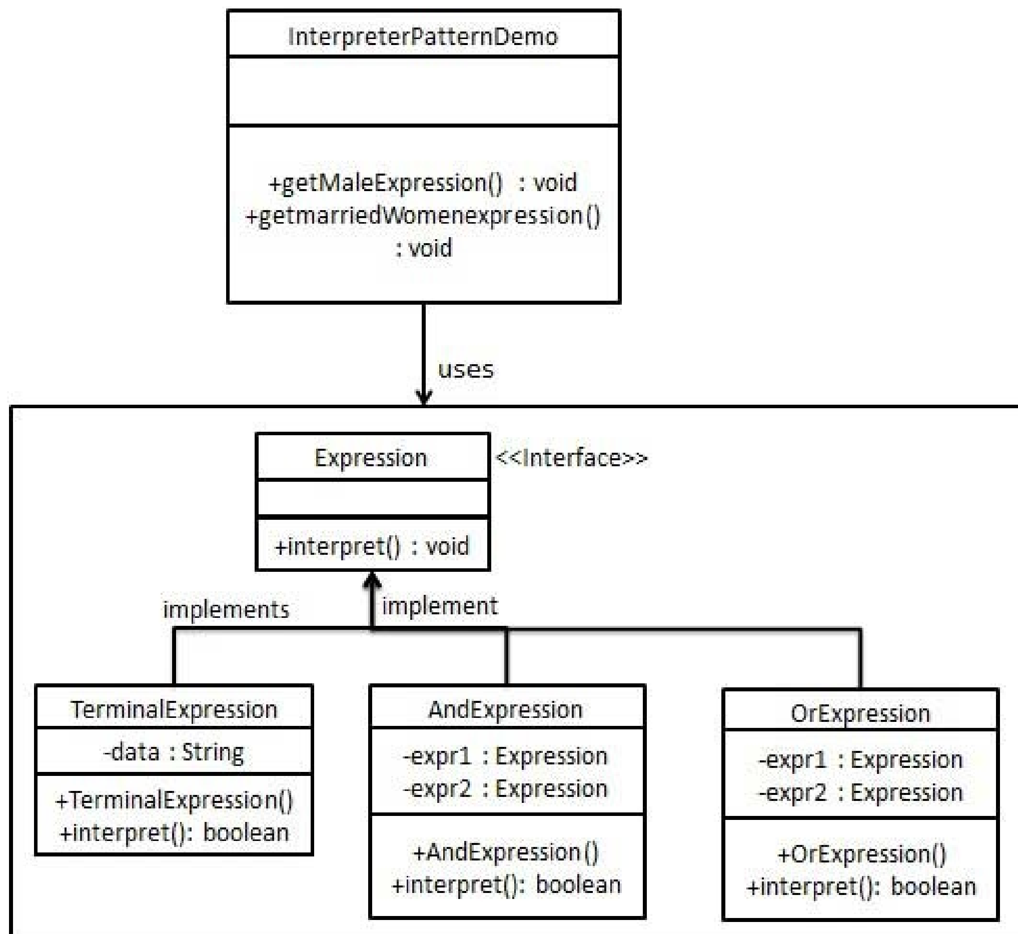
使用场景： 1、可以将一个需要解释执行的语言中的句子表示为一个抽象语法树。 2、一些重复出现的问题可以用一种简单的语言来进行表达。 3、一个简单语法需要解释的场景。

注意事项：可利用场景比较少，JAVA 中如果碰到可以用 expression4J 代替。

实现

我们将创建一个接口 *Expression* 和实现了 *Expression* 接口的实体类。定义作为上下文中主要解释器的 *TerminalExpression* 类。其他的类 *OrExpression*、*AndExpression* 用于创建组合式表达式。

InterpreterPatternDemo，我们的演示类使用 *Expression* 类创建规则和演示表达式的解析。



步骤 1

创建一个表达式接口。

Expression.java

```
public interface Expression {
   public boolean interpret(String context);
}
```

步骤 2

创建实现了上述接口的实体类。

TerminalExpression.java

```
public class TerminalExpression implements Expression {

   private String data;

   public TerminalExpression(String data){
      this.data = data;
   }

   @Override
   public boolean interpret(String context) {
      if(context.contains(data)){
         return true;
      }
      return false;
   }
}
```

OrExpression.java

```
public class OrExpression implements Expression {

   private Expression expr1 = null;
   private Expression expr2 = null;

   public OrExpression(Expression expr1, Expression expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
   }

   @Override
   public boolean interpret(String context) {
      return expr1.interpret(context) || expr2.interpret(context);
   }
}
```

AndExpression.java

```
public class AndExpression implements Expression {

   private Expression expr1 = null;
   private Expression expr2 = null;

   public AndExpression(Expression expr1, Expression expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
   }

   @Override
   public boolean interpret(String context) {
      return expr1.interpret(context) &amp;&amp; expr2.interpret(context);
   }
}
```

步骤 3

InterpreterPatternDemo 使用 *Expression* 类来创建规则，并解析它们。

InterpreterPatternDemo.java

```
public class InterpreterPatternDemo {

   //规则：Robert 和 John 是男性
   public static Expression getMaleExpression(){
      Expression robert = new TerminalExpression("Robert");
      Expression john = new TerminalExpression("John");
      return new OrExpression(robert, john);
   }

   //规则：Julie 是一个已婚的女性
   public static Expression getMarriedWomanExpression(){
      Expression julie = new TerminalExpression("Julie");
      Expression married = new TerminalExpression("Married");
      return new AndExpression(julie, married);
   }

   public static void main(String[] args) {
      Expression isMale = getMaleExpression();
      Expression isMarriedWoman = getMarriedWomanExpression();

      System.out.println("John is male? " + isMale.interpret("John"));
      System.out.println("Julie is a married women? "
      + isMarriedWoman.interpret("Married Julie"));
   }
}
```

步骤 4

验证输出。

```
John is male? true
Julie is a married women? true
```

迭代器模式

迭代器模式（Iterator Pattern）是 Java 和 .Net 编程环境中非常常用的设计模式。这种模式用于顺序访问集合对象的元素，不需要知道集合对象的底层表示。

迭代器模式属于行为型模式。

介绍

意图：提供一种方法顺序访问一个聚合对象中各个元素, 而又无须暴露该对象的内部表示。

主要解决：不同的方式来遍历整个整合对象。

何时使用：遍历一个聚合对象。

如何解决：把在元素之间游走的责任交给迭代器，而不是聚合对象。

关键代码：定义接口：hasNext, next。

应用实例：JAVA 中的 iterator。

优点： 1、它支持以不同的方式遍历一个聚合对象。 2、迭代器简化了聚合类。 3、在同一个聚合上可以有多个遍历。 4、在迭代器模式中，增加新的聚合类和迭代器类都很方便，无须修改原有代码。

缺点：由于迭代器模式将存储数据和遍历数据的职责分离，增加新的聚合类需要对应增加新的迭代器类， 类的个数成对增加， 这在一定程度上增加了系统的复杂性。

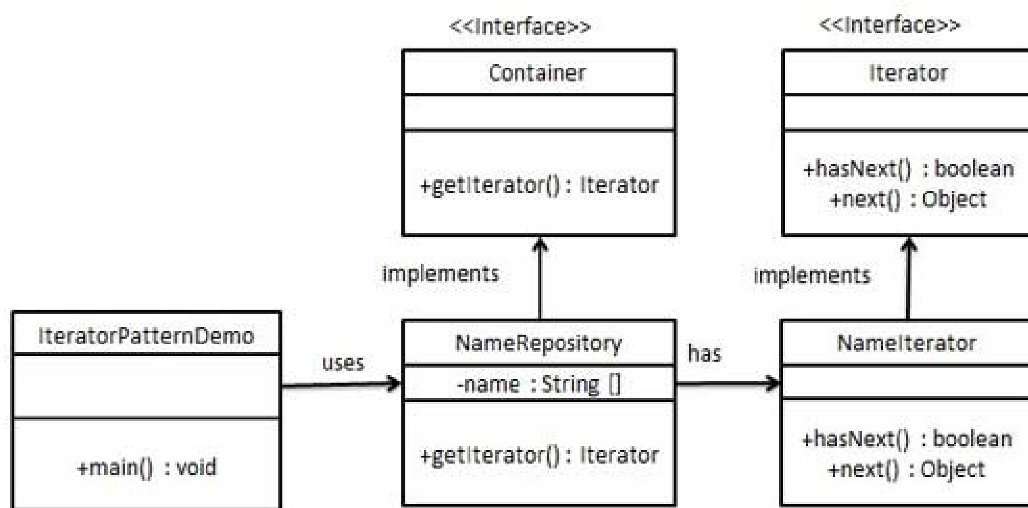
使用场景： 1、访问一个聚合对象的内容而无须暴露它的内部表示。 2、需要为聚合对象提供多种遍历方式。 3、为遍历不同的聚合结构提供一个统一的接口。

注意事项：迭代器模式就是分离了集合对象的遍历行为，抽象出一个迭代器类来负责，这样既可以做到不暴露集合的内部结构，又可让外部代码透明地访问集合内部的数据。

实现

我们将创建一个叙述导航方法的 *Iterator* 接口和一个返回迭代器的 *Container* 接口。实现了 *Container* 接口的实体类将负责实现 *Iterator* 接口。

IteratorPatternDemo，我们的演示类使用实体类 *NamesRepository* 来打印 *NamesRepository* 中存储为集合的 *Names*。



步骤 1

创建接口。

Iterator.java

```
public interface Iterator {
   public boolean hasNext();
   public Object next();
}
```

Container.java

```
public interface Container {
   public Iterator getIterator();
}
```

步骤 2

创建实现了 *Container* 接口的实体类。该类有实现了 *Iterator* 接口的内部类 *NameIterator*。

NameRepository.java

```
public class NameRepository implements Container {
   public String names[] = {"Robert" , "John" ,"Julie" , "Lora"};

   @Override
   public Iterator getIterator() {
      return new NameIterator();
   }

   private class NameIterator implements Iterator {

      int index;

      @Override
      public boolean hasNext() {
         if(index < names.length){
            return true;
         }
         return false;
      }

      @Override
      public Object next() {
         if(this.hasNext()){
            return names[index++];
         }
         return null;
      }
   }
}
```

步骤 3

使用 *NameRepository* 来获取迭代器，并打印名字。

IteratorPatternDemo.java

```
public class IteratorPatternDemo {

   public static void main(String[] args) {
      NameRepository namesRepository = new NameRepository();

      for(Iterator iter = namesRepository.getIterator(); iter.hasNext();){
         String name = (String)iter.next();
         System.out.println("Name : " + name);
      }
   }
}
```

步骤 4

验证输出。

```
Name : Robert
Name : John
Name : Julie
Name : Lora
```

中介者模式

中介者模式（Mediator Pattern）是用来降低多个对象和类之间的通信复杂性。这种模式提供了一个中介类，该类通常处理不同类之间的通信，并支持松耦合，使代码易于维护。中介者模式属于行为型模式。

介绍

意图：用一个中介对象来封装一系列的对象交互，中介者使各对象不需要显式地相互引用，从而使其耦合松散，而且可以独立地改变它们之间的交互。

主要解决：对象与对象之间存在大量的关联关系，这样势必会导致系统的结构变得很复杂，同时若一个对象发生改变，我们也需要跟踪与之相关联的对象，同时做出相应的处理。

何时使用：多个类相互耦合，形成了网状结构。

如何解决：将上述网状结构分离为星型结构。

关键代码：对象 Colleague 之间的通信封装到一个类中单独处理。

应用实例： 1、中国加入 WTO 之前是各个国家相互贸易，结构复杂，现在是各个国家通过 WTO 来互相贸易。 2、机场调度系统。 3、MVC 框架，其中C（控制器）就是 M（模型）和 V（视图）的中介者。

优点： 1、降低了类的复杂度，将一对多转化成了一对一。 2、各个类之间的解耦。 3、符合迪米特原则。

缺点：中介者会庞大，变得复杂难以维护。

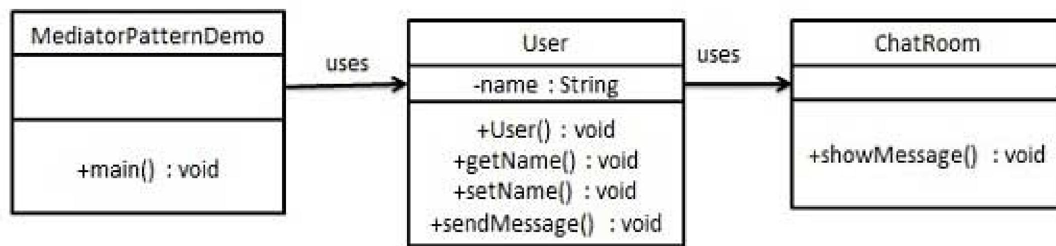
使用场景： 1、系统中对象之间存在比较复杂的引用关系，导致它们之间的依赖关系结构混乱而且难以复用该对象。 2、想通过一个中间类来封装多个类中的行为，而又不想生成太多的子类。

注意事项：不应当在职责混乱的时候使用。

实现

我们通过聊天室实例来演示中介者模式。实例中，多个用户可以向聊天室发送消息，聊天室向所有的用户显示消息。我们将创建两个类 *ChatRoom* 和 *User*。*User* 对象使用 *ChatRoom* 方法来分享他们的消息。

MediatorPatternDemo，我们的演示类使用 *User* 对象来显示他们之间的通信。



步骤 1

创建中介类。

ChatRoom.java

```
import java.util.Date;

public class ChatRoom {
   public static void showMessage(User user, String message){
      System.out.println(new Date().toString()
         + " [" + user.getName() +"] : " + message);
   }
}
```

步骤 2

创建 user 类。

User.java

```
public class User {
   private String name;

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public User(String name){
      this.name  = name;
   }

   public void sendMessage(String message){
      ChatRoom.showMessage(this,message);
   }
}
```

步骤 3

使用 *User* 对象来显示他们之间的通信。

MediatorPatternDemo.java

```
public class MediatorPatternDemo {
   public static void main(String[] args) {
      User robert = new User("Robert");
      User john = new User("John");

      robert.sendMessage("Hi! John!");
      john.sendMessage("Hello! Robert!");
   }
}
```

步骤 4

验证输出。

```
Thu Jan 31 16:05:46 IST 2013 [Robert] : Hi! John!
Thu Jan 31 16:05:46 IST 2013 [John] : Hello! Robert!
```

备忘录模式

备忘录模式（Memento Pattern）保存一个对象的某个状态，以便在适当的时候恢复对象。备忘录模式属于行为型模式。

介绍

意图：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。

主要解决：所谓备忘录模式就是在不破坏封装的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，这样可以在以后将对象恢复到原先保存的状态。

何时使用：很多时候我们总是需要记录一个对象的内部状态，这样做的目的就是为了允许用户取消不确定或者错误的操作，能够恢复到他原先的状态，使得他有"后悔药"可吃。

如何解决：通过一个备忘录类专门存储对象状态。

关键代码：客户不与备忘录类耦合，与备忘录管理类耦合。

应用实例： 1、后悔药。 2、打游戏时的存档。 3、Windows 里的 ctri + z。 4、IE 中的后退。 4、数据库的事务管理。

优点： 1、给用户提供了一种可以恢复状态的机制，可以使用户能够比较方便地回到某个历史的状态。 2、实现了信息的封装，使得用户不需要关心状态的保存细节。

缺点：消耗资源。如果类的成员变量过多，势必会占用比较大的资源，而且每一次保存都会消耗一定的内存。

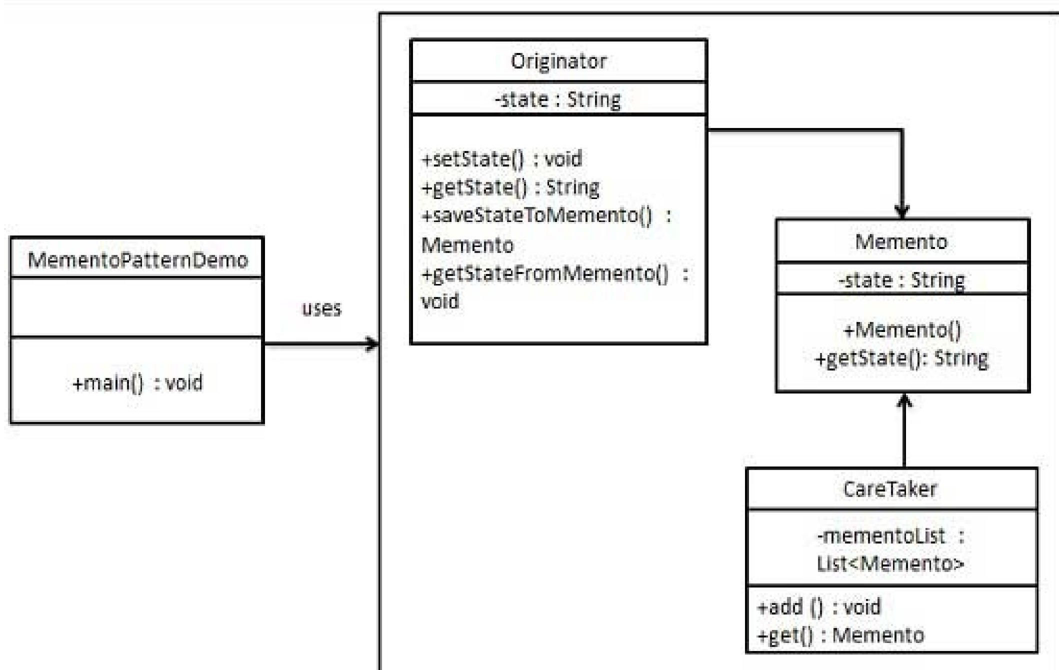
使用场景： 1、需要保存/恢复数据的相关状态场景。 2、提供一个可回滚的操作。

注意事项： 1、为了符合迪米特原则，还要增加一个管理备忘录的类。 2、为了节约内存，可使用原型模式+备忘录模式。

实现

备忘录模式使用三个类 *Memento*、*Originator* 和 *CareTaker*。Memento 包含了要被恢复的对象的状态。Originator 创建并在 Memento 对象中存储状态。Caretaker 对象负责从 Memento 中恢复对象的状态。

MementoPatternDemo，我们的演示类使用 *CareTaker* 和 *Originator* 对象来显示对象的状态恢复。



步骤 1

创建 Memento 类。

Memento.java

```
public class Memento {
   private String state;

   public Memento(String state){
      this.state = state;
   }

   public String getState(){
      return state;
   }
}
```

步骤 2

创建 Originator 类。

Originator.java

```
public class Originator {
   private String state;

   public void setState(String state){
      this.state = state;
   }

   public String getState(){
      return state;
   }

   public Memento saveStateToMemento(){
      return new Memento(state);
   }

   public void getStateFromMemento(Memento Memento){
      state = Memento.getState();
   }
}
```

步骤 3

创建 CareTaker 类。

CareTaker.java

```
import java.util.ArrayList;
import java.util.List;

public class CareTaker {
   private List<Memento> mementoList = new ArrayList<Memento>();

   public void add(Memento state){
      mementoList.add(state);
   }

   public Memento get(int index){
      return mementoList.get(index);
   }
}
```

步骤 4

使用 *CareTaker* 和 *Originator* 对象。

MementoPatternDemo.java

```
public class MementoPatternDemo {
   public static void main(String[] args) {
      Originator originator = new Originator();
      CareTaker careTaker = new CareTaker();
      originator.setState("State #1");
      originator.setState("State #2");
      careTaker.add(originator.saveStateToMemento());
      originator.setState("State #3");
      careTaker.add(originator.saveStateToMemento());
      originator.setState("State #4");

      System.out.println("Current State: " + originator.getState());
      originator.getStateFromMemento(careTaker.get(0));
      System.out.println("First saved State: " + originator.getState());
      originator.getStateFromMemento(careTaker.get(1));
      System.out.println("Second saved State: " + originator.getState());
   }
}
```

步骤 5

验证输出。

```
Current State: State #4
First saved State: State #2
Second saved State: State #3
```

观察者模式

当对象间存在一对多关系时，则使用观察者模式（Observer Pattern）。比如，当一个对象被修改时，则会自动通知它的依赖对象。观察者模式属于行为型模式。

介绍

意图：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

主要解决：一个对象状态改变给其他对象通知的问题，而且要考虑到易用和低耦合，保证高度的协作。

何时使用：一个对象（目标对象）的状态发生改变，所有的依赖对象（观察者对象）都将得到通知，进行广播通知。

如何解决：使用面向对象技术，可以将这种依赖关系弱化。

关键代码：在抽象类里有一个 ArrayList 存放观察者们。

应用实例： 1、拍卖的时候，拍卖师观察最高标价，然后通知给其他竞价者竞价。 2、西游记里面悟空请求菩萨降服红孩儿，菩萨洒了一地水招来一个老乌龟，这个乌龟就是观察者，他观察菩萨洒水这个动作。

优点： 1、观察者和被观察者是抽象耦合的。 2、建立一套触发机制。

缺点： 1、如果一个被观察者对象有很多的直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间。 2、如果在观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能导致系统崩溃。 3、观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知道观察目标发生了变化。

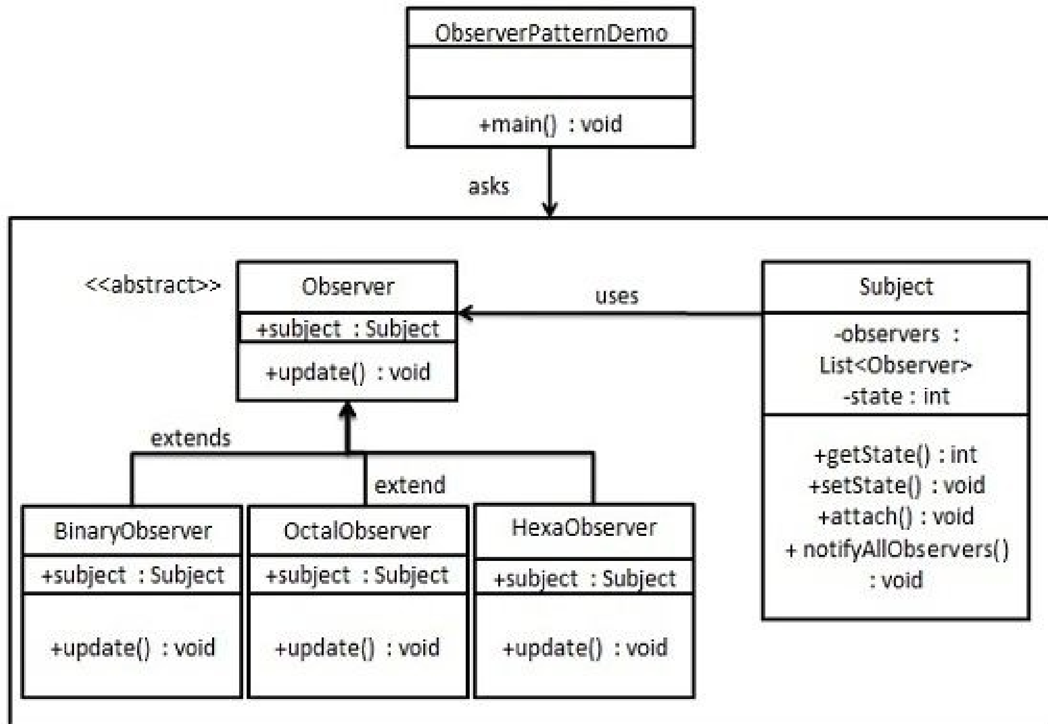
使用场景： 1、有多个子类共有的方法，且逻辑相同。 2、重要的、复杂的方法，可以考虑作为模板方法。

注意事项： 1、JAVA 中已经有了对观察者模式的支持类。 2、避免循环引用。 3、如果顺序执行，某一观察者错误会导致系统卡壳，一般采用异步方式。

实现

观察者模式使用三个类 Subject、Observer 和 Client。Subject 对象带有绑定观察者到 Client 对象和从 Client 对象解绑观察者的方法。我们创建 *Subject* 类、*Observer* 抽象类和扩展了抽象类 *Observer* 的实体类。

ObserverPatternDemo，我们的演示类使用 *Subject* 和实体类对象来演示观察者模式。



步骤 1

创建 Subject 类。

Subject.java

```
import java.util.ArrayList;
import java.util.List;

public class Subject {

   private List<Observer> observers
      = new ArrayList<Observer>();
   private int state;

   public int getState() {
      return state;
   }

   public void setState(int state) {
      this.state = state;
      notifyAllObservers();
   }

   public void attach(Observer observer){
      observers.add(observer);
   }

   public void notifyAllObservers(){
      for (Observer observer : observers) {
         observer.update();
      }
   }
}
```

步骤 2

创建 Observer 类。

Observer.java

```
public abstract class Observer {
   protected Subject subject;
   public abstract void update();
}
```

步骤 3

创建实体观察者类。

BinaryObserver.java

```
public class BinaryObserver extends Observer{

   public BinaryObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
      System.out.println( "Binary String: "
      + Integer.toBinaryString( subject.getState() ) );
   }
}
```

OctalObserver.java

```
public class OctalObserver extends Observer{

   public OctalObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
     System.out.println( "Octal String: "
     + Integer.toOctalString( subject.getState() ) );
   }
}
```

HexaObserver.java

```
public class HexaObserver extends Observer{

   public HexaObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
      System.out.println( "Hex String: "
      + Integer.toHexString( subject.getState() ).toUpperCase() );
   }
}
```

步骤 4

使用 *Subject* 和实体观察者对象。

ObserverPatternDemo.java

```
public class ObserverPatternDemo {
   public static void main(String[] args) {
      Subject subject = new Subject();

      new HexaObserver(subject);
      new OctalObserver(subject);
      new BinaryObserver(subject);

      System.out.println("First state change: 15");
      subject.setState(15);
      System.out.println("Second state change: 10");
      subject.setState(10);
   }
}
```

步骤 5

验证输出。

```
First state change: 15
Hex String: F
Octal String: 17
Binary String: 1111
Second state change: 10
Hex String: A
Octal String: 12
Binary String: 1010
```

状态模式

在状态模式（State Pattern）中，类的行为是基于它的状态改变的。这种类型的设计模式属于行为型模式。

在状态模式中，我们创建表示各种状态的对象和一个行为随着状态对象改变而改变的 context 对象。

介绍

意图：允许对象在内部状态发生改变时改变它的行为，对象看起来好像修改了它的类。

主要解决：对象的行为依赖于它的状态（属性），并且可以根据它的状态改变而改变它的相关行为。

何时使用：代码中包含大量与对象状态有关的条件语句。

如何解决：将各种具体的状态类抽象出来。

关键代码：通常命令模式的接口中只有一个方法。而状态模式的接口中有一个或者多个方法。而且，状态模式的实现类的方法，一般返回值，或者是改变实例变量的值。也就是说，状态模式一般和对象的状态有关。实现类的方法有不同的功能，覆盖接口中的方法。状态模式和命令模式一样，也可以用于消除 if...else 等条件选择语句。

应用实例： 1、打篮球的时候运动员可以有正常状态、不正常状态和超常状态。 2、曾侯乙编钟中，'钟是抽象接口','钟A'等是具体状态，'曾侯乙编钟'是具体环境（Context）。

优点： 1、封装了转换规则。 2、枚举可能的状态，在枚举状态之前需要确定状态种类。 3、将所有与某个状态有关的行为放到一个类中，并且可以方便地增加新的状态，只需要改变对象状态即可改变对象的行为。 4、允许状态转换逻辑与状态对象合成一体，而不是某一个巨大的条件语句块。 5、可以让多个环境对象共享一个状态对象，从而减少系统中对象的个数。

缺点： 1、状态模式的使用必然会增加系统类和对象的个数。 2、状态模式的结构与实现都较为复杂，如果使用不当将导致程序结构和代码的混乱。 3、状态模式对"开闭原则"的支持并不太好，对于可以切换状态的状态模式，增加新的状态类需要修改那些负责状态转换的源代码，否则无法切换到新增状态，而且修改某个状态类的行为也需修改对应类的源代码。

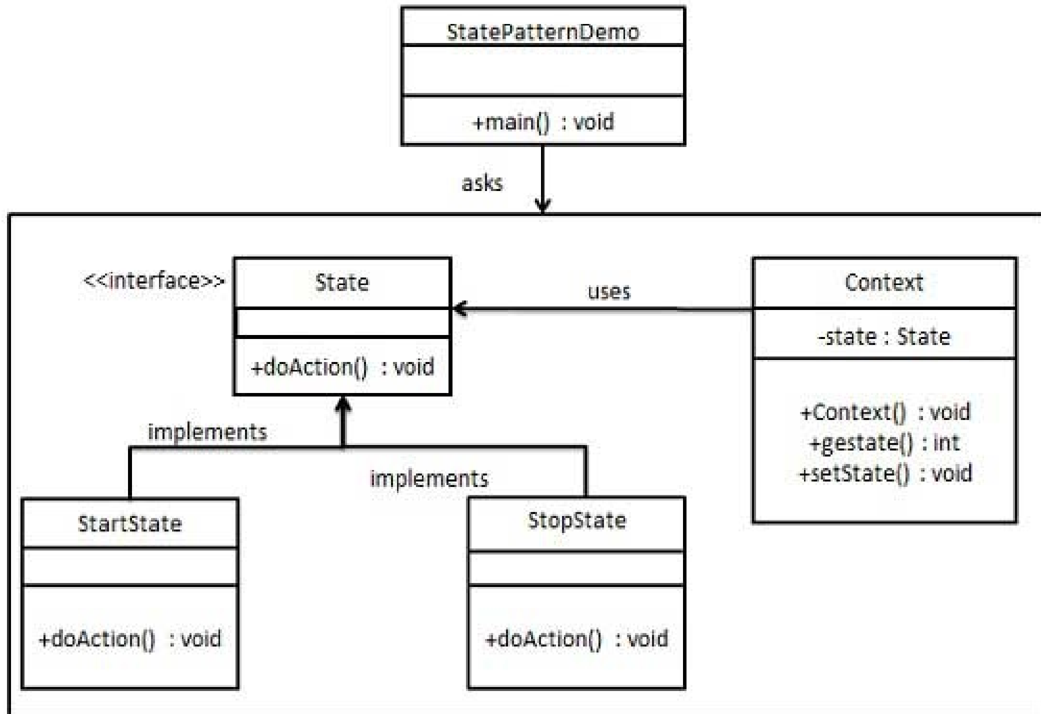
使用场景： 1、行为随状态改变而改变的场景。 2、条件、分支语句的代替者。

注意事项：在行为受状态约束的时候使用状态模式，而且状态不超过 5 个。

实现

我们将创建一个 *State* 接口和实现了 *State* 接口的实体状态类。*Context* 是一个带有某个状态的类。

StaePatternDemo，我们的演示类使用 *Context* 和状态对象来演示 Context 在状态改变时的行为变化。



步骤 1

创建一个接口。

Image.java

```
public interface State {
   public void doAction(Context context);
}
```

步骤 2

创建实现接口的实体类。

StartState.java

```
public class StartState implements State {

   public void doAction(Context context) {
      System.out.println("Player is in start state");
      context.setState(this);
   }

   public String toString(){
      return "Start State";
   }
}
```

StopState.java

```
public class StopState implements State {

   public void doAction(Context context) {
      System.out.println("Player is in stop state");
      context.setState(this);
   }

   public String toString(){
      return "Stop State";
   }
}
```

步骤 3

创建 *Context* 类。

Context.java

```
public class Context {
   private State state;

   public Context(){
      state = null;
   }

   public void setState(State state){
      this.state = state;
   }

   public State getState(){
      return state;
   }
}
```

步骤 4

使用 *Context* 来查看当状态 *State* 改变时的行为变化。

StatePatternDemo.java

```
public class StatePatternDemo {
   public static void main(String[] args) {
      Context context = new Context();

      StartState startState = new StartState();
      startState.doAction(context);

      System.out.println(context.getState().toString());

      StopState stopState = new StopState();
      stopState.doAction(context);

      System.out.println(context.getState().toString());
   }
}
```

步骤 5

验证输出。

```
Player is in start state
Start State
Player is in stop state
Stop State
```

空对象模式

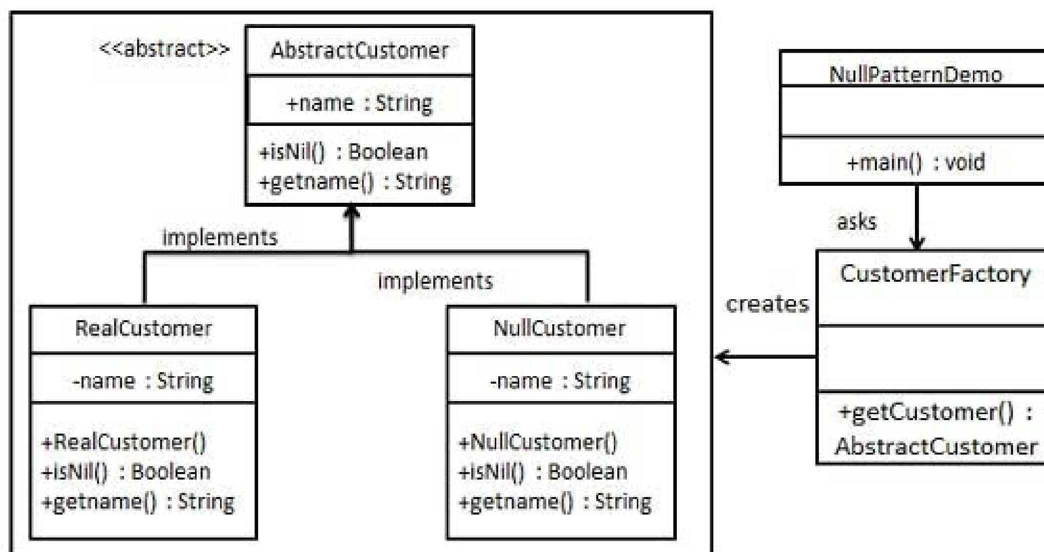
在空对象模式（Null Object Pattern）中，一个空对象取代 NULL 对象实例的检查。Null 对象不是检查空值，而是反应一个不做任何动作的关系。这样的 Null 对象也可以在数据不可用的时候提供默认的行为。

在空对象模式中，我们创建一个指定各种要执行的操作的抽象类和扩展该类的实体类，还创建一个未对该类做任何实现的空对象类，该空对象类将无缝地使用在需要检查空值的地方。

实现

我们将创建一个定义操作（在这里，是客户的名称）的 *AbstractCustomer* 抽象类，和扩展了 *AbstractCustomer* 类的实体类。工厂类 *CustomerFactory* 基于客户传递的名字来返回 *RealCustomer* 或 *NullCustomer* 对象。

NullPatternDemo，我们的演示类使用 *CustomerFactory* 来演示空对象模式的用法。



步骤 1

创建一个抽象类。

AbstractCustomer.java

```
public abstract class AbstractCustomer {
   protected String name;
   public abstract boolean isNil();
   public abstract String getName();
}
```

步骤 2

创建扩展了上述类的实体类。

RealCustomer.java

```
public class RealCustomer extends AbstractCustomer {

   public RealCustomer(String name) {
      this.name = name;
   }

   @Override
   public String getName() {
      return name;
   }

   @Override
   public boolean isNil() {
      return false;
   }
}
```

NullCustomer.java

```
public class NullCustomer extends AbstractCustomer {

   @Override
   public String getName() {
      return "Not Available in Customer Database";
   }

   @Override
   public boolean isNil() {
      return true;
   }
}
```

步骤 3

创建 *CustomerFactory* 类。

CustomerFactory.java

```
public class CustomerFactory {

   public static final String[] names = {"Rob", "Joe", "Julie"};

   public static AbstractCustomer getCustomer(String name){
      for (int i = 0; i < names.length; i++) {
         if (names[i].equalsIgnoreCase(name)){
            return new RealCustomer(name);
         }
      }
      return new NullCustomer();
   }
}
```

步骤 4

使用 *CustomerFactory*，基于客户传递的名字，来获取 *RealCustomer* 或 *NullCustomer* 对象。

NullPatternDemo.java

```
public class NullPatternDemo {
   public static void main(String[] args) {

      AbstractCustomer customer1 = CustomerFactory.getCustomer("Rob");
      AbstractCustomer customer2 = CustomerFactory.getCustomer("Bob");
      AbstractCustomer customer3 = CustomerFactory.getCustomer("Julie");
      AbstractCustomer customer4 = CustomerFactory.getCustomer("Laura");

      System.out.println("Customers");
      System.out.println(customer1.getName());
      System.out.println(customer2.getName());
      System.out.println(customer3.getName());
      System.out.println(customer4.getName());
   }
}
```

步骤 5

验证输出。

```
Customers
Rob
Not Available in Customer Database
Julie
Not Available in Customer Database
```

策略模式

在策略模式（Strategy Pattern）中，一个类的行为或其算法可以在运行时更改。这种类型的设计模式属于行为型模式。

在策略模式中，我们创建表示各种策略的对象和一个行为随着策略对象改变而改变的 context 对象。策略对象改变 context 对象的执行算法。

介绍

意图：定义一系列的算法,把它们一个个封装起来, 并且使它们可相互替换。

主要解决：在有多种算法相似的情况下，使用 if...else 所带来的复杂和难以维护。

何时使用：一个系统有许多许多类，而区分它们的只是他们直接的行为。

如何解决：将这些算法封装成一个一个的类，任意地替换。

关键代码：实现同一个接口。

应用实例： 1、诸葛亮的锦囊妙计，每一个锦囊就是一个策略。 2、旅行的出游方式，选择骑自行车、坐汽车，每一种旅行方式都是一个策略。 3、JAVA AWT 中的 LayoutManager。

优点： 1、算法可以自由切换。 2、避免使用多重条件判断。 3、扩展性良好。

缺点： 1、策略类会增多。 2、所有策略类都需要对外暴露。

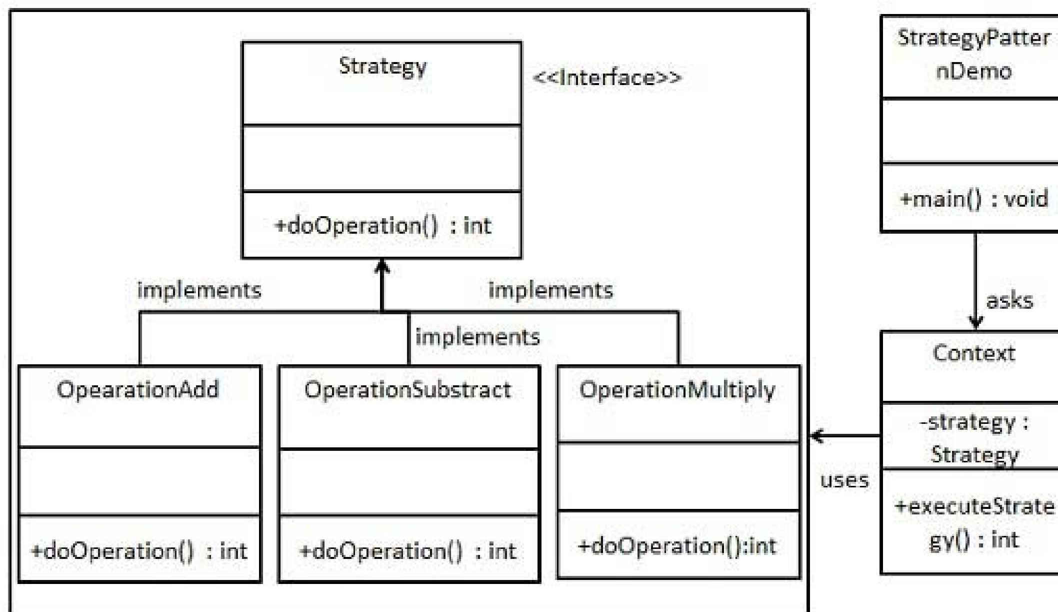
使用场景： 1、如果在一个系统里面有许多类，它们之间的区别仅在于它们的行为，那么使用策略模式可以动态地让一个对象在许多行为中选择一种行为。 2、一个系统需要动态地在几种算法中选择一种。 3、如果一个对象有很多的行为，如果不用恰当的模式，这些行为就只好使用多重的条件选择语句来实现。

注意事项：如果一个系统的策略多于四个，就需要考虑使用混合模式，解决策略类膨胀的问题。

实现

我们将创建一个定义活动的 *Strategy* 接口和实现了 *Strategy* 接口的实体策略类。*Context* 是一个使用了某种策略的类。

StrategyPatternDemo，我们的演示类使用 *Context* 和策略对象来演示 Context 在它所配置或使用的策略改变时的行为变化。



步骤 1

创建一个接口。

Strategy.java

```
public interface Strategy {
   public int doOperation(int num1, int num2);
}
```

步骤 2

创建实现接口的实体类。

OperationAdd.java

```
public class OperationAdd implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 + num2;
   }
}
```

OperationSubstract.java

```
public class OperationSubstract implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 - num2;
   }
}
```

OperationMultiply.java

```
public class OperationMultiply implements Strategy{
   @Override
   public int doOperation(int num1, int num2) {
      return num1 * num2;
   }
}
```

步骤 3

创建 *Context* 类。

Context.java

```
public class Context {
   private Strategy strategy;

   public Context(Strategy strategy){
      this.strategy = strategy;
   }

   public int executeStrategy(int num1, int num2){
      return strategy.doOperation(num1, num2);
   }
}
```

步骤 4

使用 *Context* 来查看当它改变策略 *Strategy* 时的行为变化。

StatePatternDemo.java

```
public class StrategyPatternDemo {
   public static void main(String[] args) {
      Context context = new Context(new OperationAdd());
      System.out.println("10 + 5 = " + context.executeStrategy(10, 5));

      context = new Context(new OperationSubstract());
      System.out.println("10 - 5 = " + context.executeStrategy(10, 5));

      context = new Context(new OperationMultiply());
      System.out.println("10 * 5 = " + context.executeStrategy(10, 5));
   }
}
```

步骤 5

验证输出。

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
```

模板模式

在模板模式（Template Pattern）中，一个抽象类公开定义了执行它的方法的方式/模板。它的子类可以按需要重写方法实现，但调用将以抽象类中定义的方式进行。这种类型的设计模式属于行为型模式。

介绍

意图：定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。

主要解决：一些方法通用，却在每一个子类都重新写了这一方法。

何时使用：有一些通用的方法。

如何解决：将这些通用算法抽象出来。

关键代码：在抽象类实现，其他步骤在子类实现。

应用实例： 1、在造房子的时候，地基、走线、水管都一样，只有在建筑的后期才有加壁橱加栅栏等差异。 2、西游记里面菩萨定好的 81 难，这就是一个顶层的逻辑骨架。 3、Spirng 中对 Hibernate 的支持，将一些已经定好的方法封装起来，比如开启事务、获取 Session、关闭 Session 等，程序员不重复写那些已经规范好的代码，直接丢一个实体就可以保存。

优点： 1、封装不变部分，扩展可变部分。 2、提取公共代码，便于维护。 3、行为由父类控制，子类实现。

缺点：每一个不同的实现都需要一个子类来实现，导致类的个数增加，使得系统更加庞大。

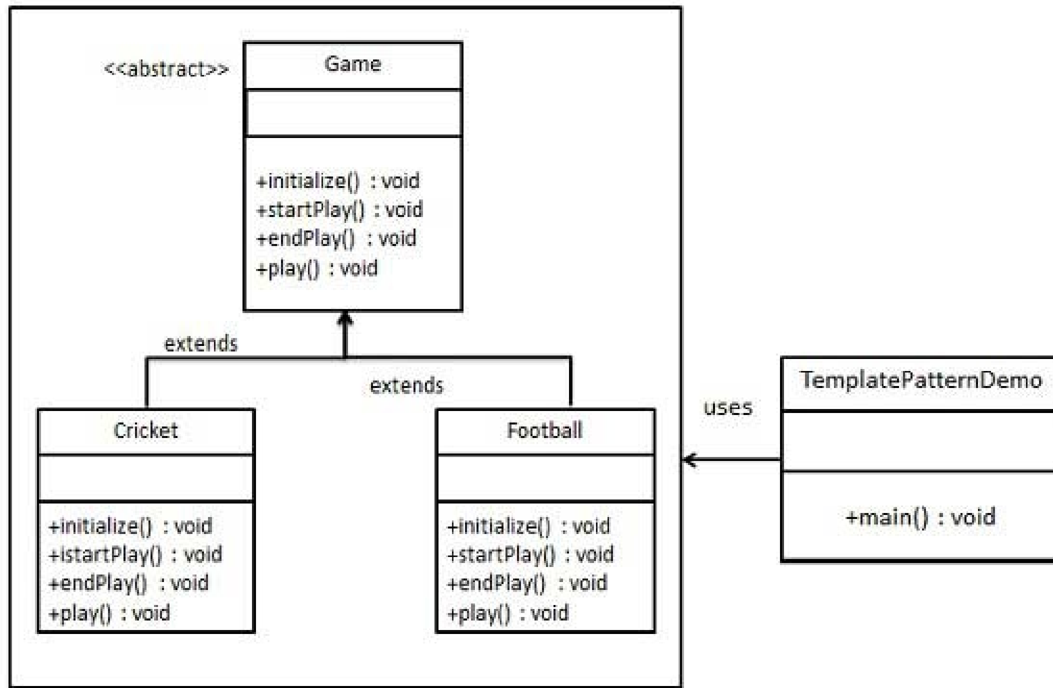
使用场景： 1、有多个子类共有的方法，且逻辑相同。 2、重要的、复杂的方法，可以考虑作为模板方法。

注意事项：为防止恶意操作，一般模板方法都加上 final 关键词。

实现

我们将创建一个定义操作的 *Game* 抽象类，其中，模板方法设置为 final，这样它就不会被重写。*Cricket* 和 *Football* 是扩展了 *Game* 的实体类，它们重写了抽象类的方法。

TemplatePatternDemo，我们的演示类使用 *Game* 来演示模板模式的用法。



步骤 1

创建一个抽象类，它的模板方法被设置为 final。

Game.java

```
public abstract class Game {
   abstract void initialize();
   abstract void startPlay();
   abstract void endPlay();

   //模板
   public final void play(){

      //初始化游戏
      initialize();

      //开始游戏
      startPlay();

      //结束游戏
      endPlay();
   }
}
```

步骤 2

创建扩展了上述类的实体类。

Cricket.java

```
public class Cricket extends Game {

   @Override
   void endPlay() {
      System.out.println("Cricket Game Finished!");
   }

   @Override
   void initialize() {
      System.out.println("Cricket Game Initialized! Start playing.");
   }

   @Override
   void startPlay() {
      System.out.println("Cricket Game Started. Enjoy the game!");
   }
}
```

Football.java

```
public class Football extends Game {

   @Override
   void endPlay() {
      System.out.println("Football Game Finished!");
   }

   @Override
   void initialize() {
      System.out.println("Football Game Initialized! Start playing.");
   }

   @Override
   void startPlay() {
      System.out.println("Football Game Started. Enjoy the game!");
   }
}
```

步骤 3

使用 *Game* 的模板方法 play() 来演示游戏的定义方式。

TemplatePatternDemo.java

```
public class TemplatePatternDemo {
   public static void main(String[] args) {

      Game game = new Cricket();
      game.play();
      System.out.println();
      game = new Football();
      game.play();
   }
}
```

步骤 4

验证输出。

```
Cricket Game Initialized! Start playing.
Cricket Game Started. Enjoy the game!
Cricket Game Finished!

Football Game Initialized! Start playing.
Football Game Started. Enjoy the game!
Football Game Finished!
```

访问者模式

在访问者模式（Visitor Pattern）中，我们使用了一个访问者类，它改变了元素类的执行算法。通过这种方式，元素的执行算法可以随着访问者改变而改变。这种类型的设计模式属于行为型模式。根据模式，元素对象已接受访问者对象，这样访问者对象就可以处理元素对象上的操作。

介绍

意图：主要将数据结构与数据操作分离。

主要解决：稳定的数据结构和易变的操作耦合问题。

何时使用：需要对一个对象结构中的对象进行很多不同的并且不相关的操作，而需要避免让这些操作"污染"这些对象的类，使用访问者模式将这些封装到类中。

如何解决：在被访问的类里面加一个对外提供接待访问者的接口。

关键代码：在数据基础类里面有一个方法接受访问者，将自身引用传入访问者。

应用实例：您在朋友家做客，您是访问者，朋友接受您的访问，您通过朋友的描述，然后对朋友的描述做出一个判断，这就是访问者模式。

优点： 1、符合单一职责原则。 2、优秀的扩展性。 3、灵活性。

缺点： 1、具体元素对访问者公布细节，违反了迪米特原则。 2、具体元素变更比较困难。 3、违反了依赖倒置原则，依赖了具体类，没有依赖抽象。

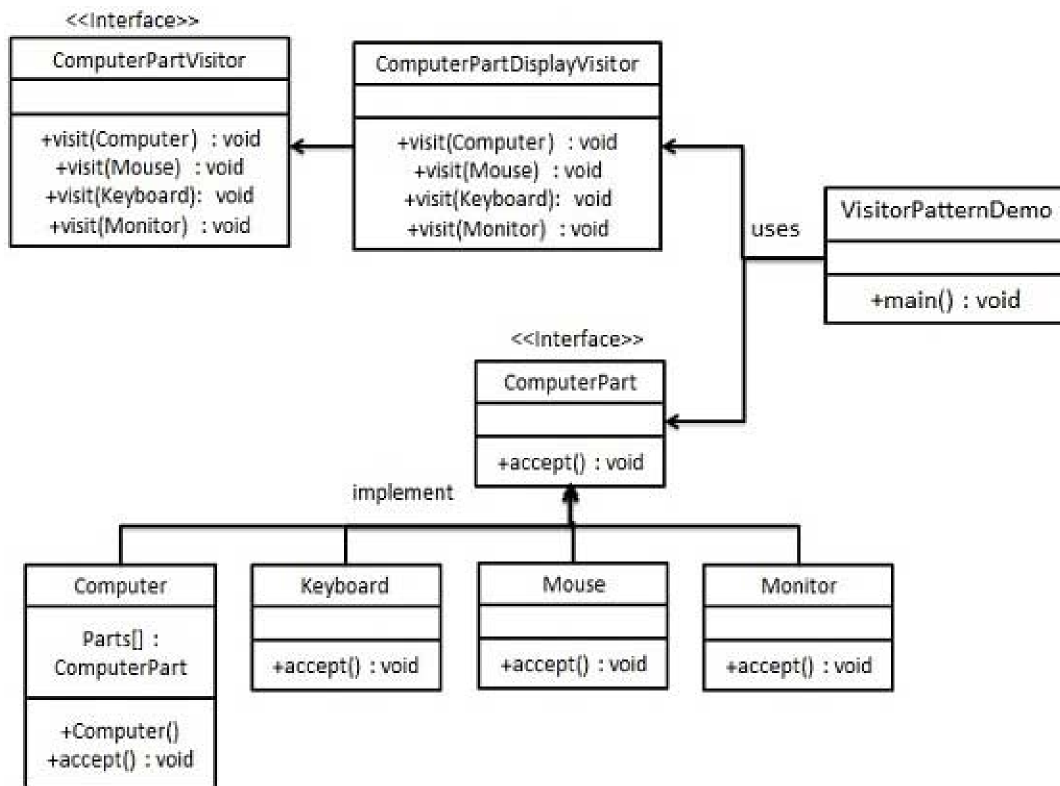
使用场景： 1、对象结构中对象对应的类很少改变，但经常需要在此对象结构上定义新的操作。 2、需要对一个对象结构中的对象进行很多不同的并且不相关的操作，而需要避免让这些操作"污染"这些对象的类，也不希望在增加新操作时修改这些类。

注意事项：访问者可以对功能进行统一，可以做报表、UI、拦截器与过滤器。

实现

我们将创建一个定义接受操作的 *ComputerPart* 接口。*Keyboard*、*Mouse*、*Monitor* 和 *Computer* 是实现了 *ComputerPart* 接口的实体类。我们将定义另一个接口 *ComputerPartVisitor*，它定义了访问者类的操作。*Computer* 使用实体访问者来执行相应的动作。

VisitorPatternDemo，我们的演示类使用 *Computer*、*ComputerPartVisitor* 类来演示访问者模式的用法。



步骤 1

定义一个表示元素的接口。

ComputerPart.java

```
public interface class ComputerPart {
   public void accept(ComputerPartVisitor computerPartVisitor);
}
```

步骤 2

创建扩展了上述类的实体类。

Keyboard.java

```
public class Keyboard  implements ComputerPart {

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      computerPartVisitor.visit(this);
   }
}
```

Monitor.java

```
public class Monitor  implements ComputerPart {

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      computerPartVisitor.visit(this);
   }
}
```

Mouse.java

```
public class Mouse  implements ComputerPart {

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      computerPartVisitor.visit(this);
   }
}
```

Computer.java

```
public class Computer implements ComputerPart {

   ComputerPart[] parts;

   public Computer(){
      parts = new ComputerPart[] {new Mouse(), new Keyboard(), new Monitor()};
   }

   @Override
   public void accept(ComputerPartVisitor computerPartVisitor) {
      for (int i = 0; i < parts.length; i++) {
         parts[i].accept(computerPartVisitor);
      }
      computerPartVisitor.visit(this);
   }
}
```

步骤 3

定义一个表示访问者的接口。

ComputerPartVisitor.java

```
public interface ComputerPartVisitor {
    public void visit(Computer computer);
    public void visit(Mouse mouse);
    public void visit(Keyboard keyboard);
    public void visit(Monitor monitor);
}
```

步骤 4

创建实现了上述类的实体访问者。

ComputerPartDisplayVisitor.java

```
public class ComputerPartDisplayVisitor implements ComputerPartVisitor {

   @Override
   public void visit(Computer computer) {
      System.out.println("Displaying Computer.");
   }

   @Override
   public void visit(Mouse mouse) {
      System.out.println("Displaying Mouse.");
   }

   @Override
   public void visit(Keyboard keyboard) {
      System.out.println("Displaying Keyboard.");
   }

   @Override
   public void visit(Monitor monitor) {
      System.out.println("Displaying Monitor.");
   }
}
```

步骤 5

使用 *ComputerPartDisplayVisitor* 来显示 *Computer* 的组成部分。

VisitorPatternDemo.java

```
public class VisitorPatternDemo {
   public static void main(String[] args) {

      ComputerPart computer = new Computer();
      computer.accept(new ComputerPartDisplayVisitor());
   }
}
```

步骤 6

验证输出。

```
Displaying Mouse.
Displaying Keyboard.
Displaying Monitor.
Displaying Computer.
```

MVC 模式

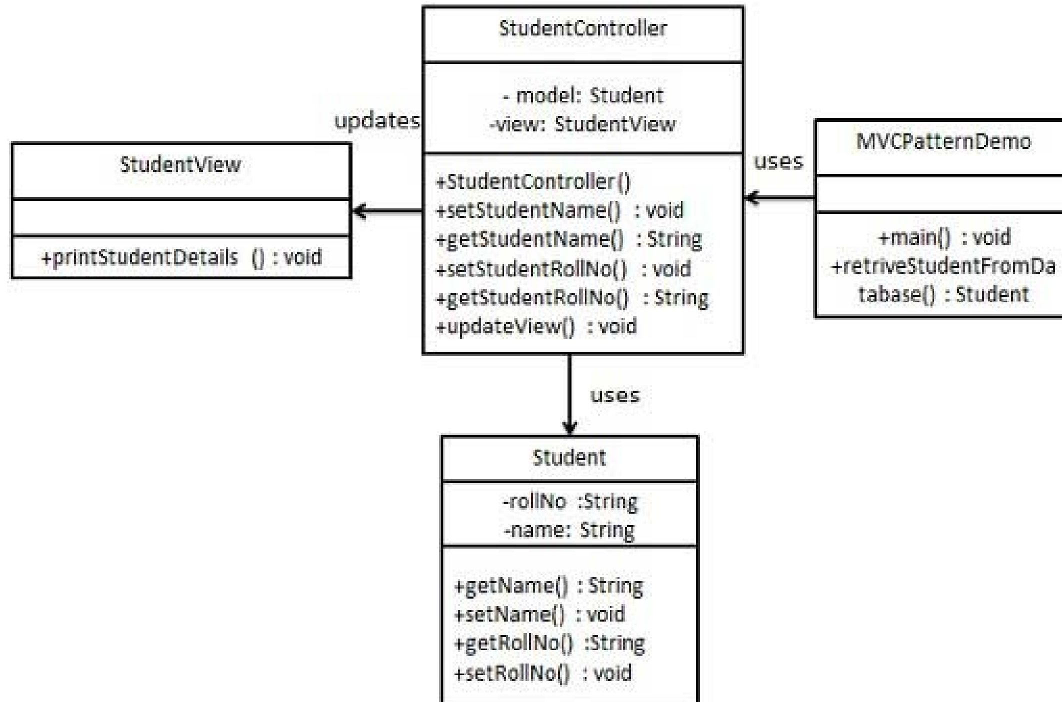
MVC 模式代表 Model-View-Controller（模型-视图-控制器） 模式。这种模式用于应用程序的分层开发。

- **Model**（模型） - 模型代表一个存取数据的对象或 JAVA POJO。它也可以带有逻辑，在数据变化时更新控制器。
- **View**（视图） - 视图代表模型包含的数据的可视化。
- **Controller**（控制器） - 控制器作用于模型和视图上。它控制数据流向模型对象，并在数据变化时更新视图。它使视图与模型分离开。

实现

我们将创建一个作为模型的 *Student* 对象。*StudentView* 是一个把学生详细信息输出到控制台的视图类，*StudentController* 是负责存储数据到 *Student* 对象中的控制器类，并相应地更新视图 *StudentView*。

MVCPatternDemo，我们的演示类使用 *StudentController* 来演示 MVC 模式的用法。



步骤 1

创建模型。

Student.java

```
public class Student {
   private String rollNo;
   private String name;
   public String getRollNo() {
      return rollNo;
   }
   public void setRollNo(String rollNo) {
      this.rollNo = rollNo;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

步骤 2

创建视图。

StudentView.java

```
public class StudentView {
   public void printStudentDetails(String studentName, String studentRollNo){
      System.out.println("Student: ");
      System.out.println("Name: " + studentName);
      System.out.println("Roll No: " + studentRollNo);
   }
}
```

步骤 3

创建控制器。

StudentController.java

```
public class StudentController {
   private Student model;
   private StudentView view;

   public StudentController(Student model, StudentView view){
      this.model = model;
      this.view = view;
   }

   public void setStudentName(String name){
      model.setName(name);
   }

   public String getStudentName(){
      return model.getName();
   }

   public void setStudentRollNo(String rollNo){
      model.setRollNo(rollNo);
   }

   public String getStudentRollNo(){
      return model.getRollNo();
   }

   public void updateView(){
      view.printStudentDetails(model.getName(), model.getRollNo());
   }
}
```

步骤 4

使用 *StudentController* 方法来演示 MVC 设计模式的用法。

MVCPatternDemo.java

```
public class MVCPatternDemo {
   public static void main(String[] args) {

      //从数据可获取学生记录
      Student model  = retriveStudentFromDatabase();

      //创建一个视图：把学生详细信息输出到控制台
      StudentView view = new StudentView();

      StudentController controller = new StudentController(model, view);

      controller.updateView();

      //更新模型数据
      controller.setStudentName("John");

      controller.updateView();
   }

   private static Student retriveStudentFromDatabase(){
      Student student = new Student();
      student.setName("Robert");
      student.setRollNo("10");
      return student;
   }
}
```

步骤 5

验证输出。

```
Student:
Name: Robert
Roll No: 10
Student:
Name: John
Roll No: 10
```

业务代表模式

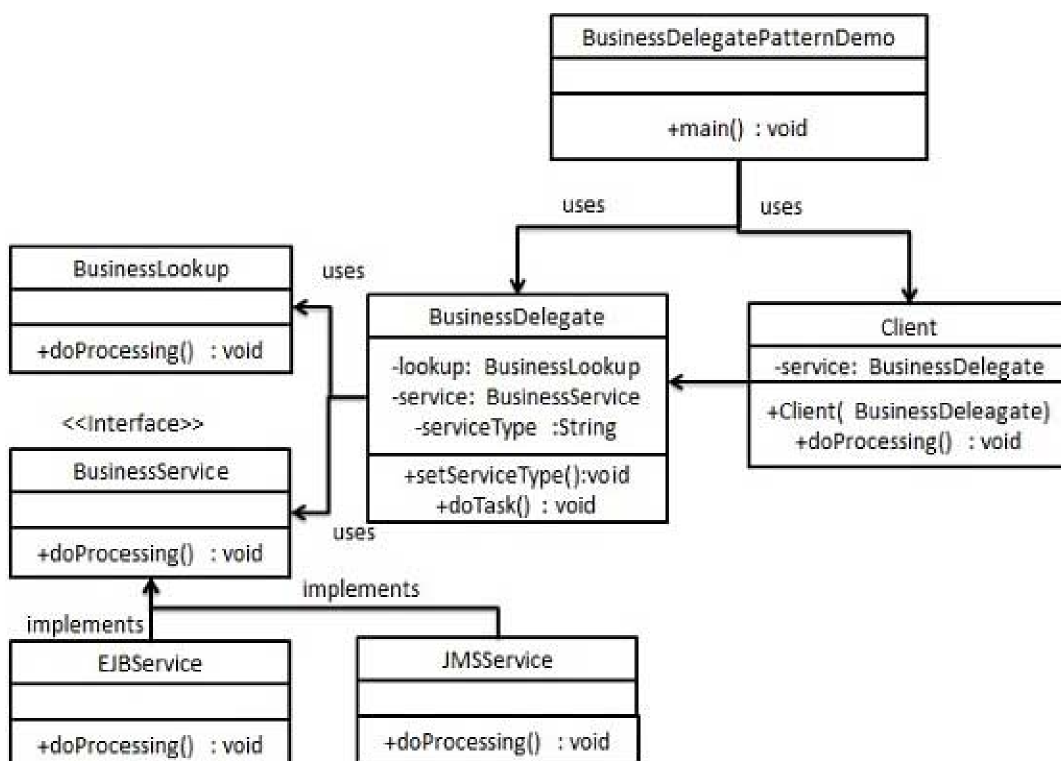
业务代表模式（Business Delegate Pattern）用于对表示层和业务层解耦。它基本上是用来减少通信或对表示层代码中的业务层代码的远程查询功能。在业务层中我们有以下实体。

- 客户端（**Client**） - 表示层代码可以是 JSP、servlet 或 UI java 代码。
- 业务代表（**Business Delegate**） - 一个为客户端实体提供的入口类，它提供了对业务服务方法的访问。
- 查询服务（**LookUp Service**） - 查找服务对象负责获取相关的业务实现，并提供业务对象对业务代表对象的访问。
- 业务服务（**Business Service**） - 业务服务接口。实现了该业务服务的实体类，提供了实际的业务实现逻辑。

实现

我们将创建 *Client*、*BusinessDelegate*、*BusinessService*、*LookUpService*、*JMSService* 和 *EJBService* 来表示业务代表模式中的各种实体。

BusinessDelegatePatternDemo，我们的演示类使用 *BusinessDelegate* 和 *Client* 来演示业务代表模式的用法。



步骤 1

创建 BusinessService 接口。

BusinessService.java

```
public interface BusinessService {
   public void doProcessing();
}
```

步骤 2

创建实体服务类。

EJBService.java

```
public class EJBService implements BusinessService {

   @Override
   public void doProcessing() {
      System.out.println("Processing task by invoking EJB Service");
   }
}
```

JMSService.java

```
public class JMSService implements BusinessService {

   @Override
   public void doProcessing() {
      System.out.println("Processing task by invoking JMS Service");
   }
}
```

步骤 3

创建业务查询服务。

BusinessLookUp.java

```
public class BusinessLookUp {
   public BusinessService getBusinessService(String serviceType){
      if(serviceType.equalsIgnoreCase("EJB")){
         return new EJBService();
      }else {
         return new JMSService();
      }
   }
}
```

步骤 4

创建业务代表。

BusinessDelegate.java

```
public class BusinessDelegate {
   private BusinessLookUp lookupService = new BusinessLookUp();
   private BusinessService businessService;
   private String serviceType;

   public void setServiceType(String serviceType){
      this.serviceType = serviceType;
   }

   public void doTask(){
      businessService = lookupService.getBusinessService(serviceType);
      businessService.doProcessing();
   }
}
```

步骤 5

创建客户端。

Student.java

```
public class Client {

   BusinessDelegate businessService;

   public Client(BusinessDelegate businessService){
      this.businessService  = businessService;
   }

   public void doTask(){
      businessService.doTask();
   }
}
```

步骤 6

使用 BusinessDelegate 和 Client 类来演示业务代表模式。

BusinessDelegatePatternDemo.java

```
public class BusinessDelegatePatternDemo {

   public static void main(String[] args) {

      BusinessDelegate businessDelegate = new BusinessDelegate();
      businessDelegate.setServiceType("EJB");

      Client client = new Client(businessDelegate);
      client.doTask();

      businessDelegate.setServiceType("JMS");
      client.doTask();
   }
}
```

步骤 7

验证输出。

```
Processing task by invoking EJB Service
Processing task by invoking JMS Service
```

组合实体模式

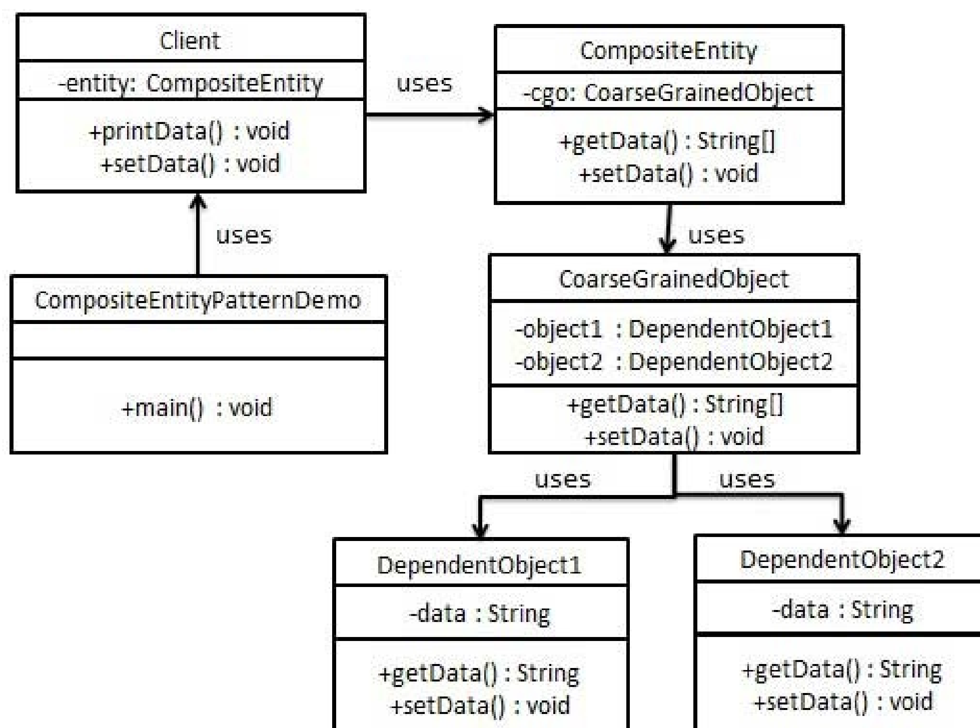
组合实体模式（Composite Entity Pattern）用在 EJB 持久化机制中。一个组合实体是一个 EJB 实体 bean，代表了对象的图解。当更新一个组合实体时，内部依赖对象 beans 会自动更新，因为它们是由 EJB 实体 bean 管理的。以下是组合实体 bean 的参与者。

- 组合实体（**Composite Entity**） - 它是主要的实体 bean。它可以是粗粒的，或者可以包含一个粗粒度对象，用于持续生命周期。
- 粗粒度对象（**Coarse-Grained Object**） - 该对象包含以来对象。它有自己的生命周期，也能管理依赖对象的生命周期。
- 依赖对象（**Dependent Object**） - 依赖对象是一个持续生命周期依赖于粗粒度对象的对象。
- 策略（**Strategies**） - 策略表示如何实现组合实体。

实现

我们将创建作为组合实体的 *CompositeEntity* 对象。*CoarseGrainedObject* 是一个包含依赖对象的类。

CompositeEntityPatternDemo，我们的演示类使用 *Client* 类来演示组合实体模式的用法。



步骤 1

创建依赖对象。

DependentObject1.java

```
public class DependentObject1 {

   private String data;

   public void setData(String data){
      this.data = data;
   }

   public String getData(){
      return data;
   }
}
```

DependentObject2.java

```
public class DependentObject2 {

   private String data;

   public void setData(String data){
      this.data = data;
   }

   public String getData(){
      return data;
   }
}
```

步骤 2

创建粗粒度对象。

CoarseGrainedObject.java

```
public class CoarseGrainedObject {
   DependentObject1 do1 = new DependentObject1();
   DependentObject2 do2 = new DependentObject2();

   public void setData(String data1, String data2){
      do1.setData(data1);
      do2.setData(data2);
   }

   public String[] getData(){
      return new String[] {do1.getData(),do2.getData()};
   }
}
```

步骤 3

创建组合实体。

CompositeEntity.java

```
public class CompositeEntity {
   private CoarseGrainedObject cgo = new CoarseGrainedObject();

   public void setData(String data1, String data2){
      cgo.setData(data1, data2);
   }

   public String[] getData(){
      return cgo.getData();
   }
}
```

步骤 4

创建使用组合实体的客户端类。

Client.java

```
public class Client {
   private CompositeEntity compositeEntity = new CompositeEntity();

   public void printData(){
      for (int i = 0; i < compositeEntity.getData().length; i++) {
         System.out.println("Data: " + compositeEntity.getData()[i]);
      }
   }

   public void setData(String data1, String data2){
      compositeEntity.setData(data1, data2);
   }
}
```

步骤 5

使用 *Client* 来演示组合实体设计模式的用法。

CompositeEntityPatternDemo.java

```
public class CompositeEntityPatternDemo {
   public static void main(String[] args) {
       Client client = new Client();
       client.setData("Test", "Data");
       client.printData();
       client.setData("Second Test", "Data1");
       client.printData();
   }
}
```

步骤 6

验证输出。

```
Data: Test
Data: Data
Data: Second Test
Data: Data1
```

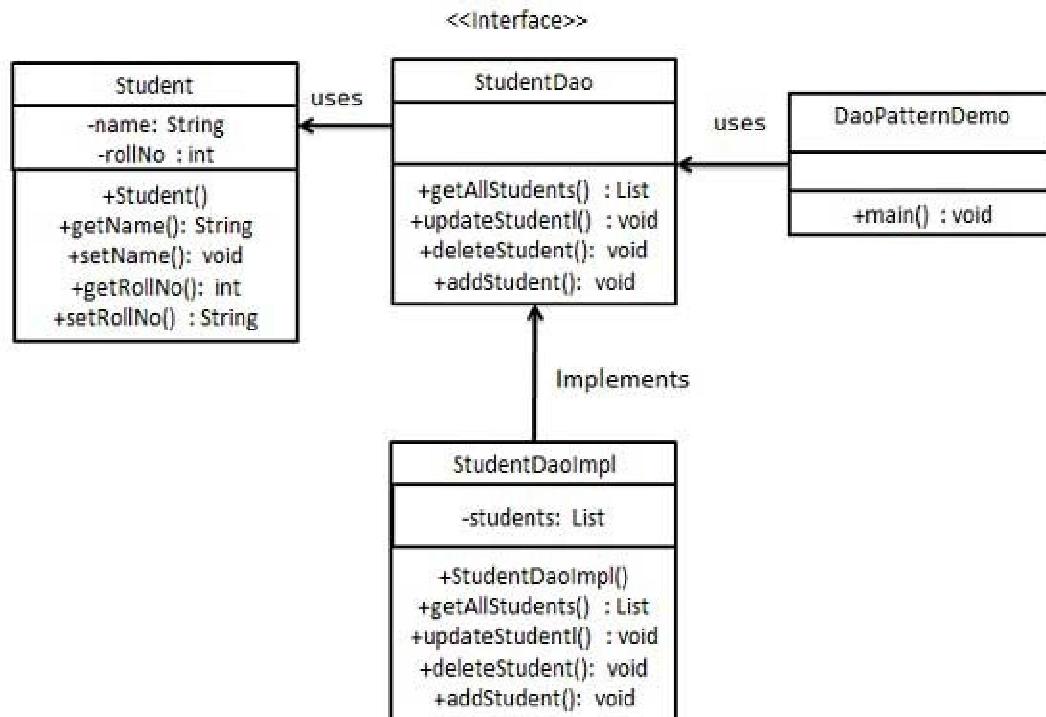
数据访问对象模式

数据访问对象模式（Data Access Object Pattern）或 DAO 模式用于把低级的数据访问 API 或操作从高级的业务服务中分离出来。以下是数据访问对象模式的参与者。

- 数据访问对象接口（**Data Access Object Interface**） - 该接口定义了在一个模型对象上要执行的标准操作。
- 数据访问对象实体类（**Data Access Object concrete class**） - 该类实现了上述的接口。该类负责从数据源获取数据，数据源可以是数据库，也可以是 xml，或者是其他的存储机制。
- 模型对象/数值对象（**Model Object/Value Object**） - 该对象是简单的 POJO，包含了 get/set 方法来存储通过使用 DAO 类检索到的数据。

实现

我们将创建一个作为模型对象或数值对象的 *Student* 对象。*StudentDao* 是数据访问对象接口。*StudentDaoImpl* 是实现了数据访问对象接口的实体类。*DaoPatternDemo*，我们的演示类使用 *StudentDao* 来演示数据访问对象模式的用法。



步骤 1

创建数值对象。

Student.java

```
public class Student {
   private String name;
   private int rollNo;

   Student(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getRollNo() {
      return rollNo;
   }

   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
}
```

步骤 2

创建数据访问对象接口。

StudentDao.java

```
import java.util.List;

public interface StudentDao {
   public List<Student> getAllStudents();
   public Student getStudent(int rollNo);
   public void updateStudent(Student student);
   public void deleteStudent(Student student);
}
```

步骤 3

创建实现了上述接口的实体类。

StudentDaoImpl.java

```
import java.util.ArrayList;
import java.util.List;

public class StudentDaoImpl implements StudentDao {

   //列表是当作一个数据库
   List<Student> students;

   public StudentDaoImpl(){
      students = new ArrayList<Student>();
      Student student1 = new Student("Robert",0);
      Student student2 = new Student("John",1);
      students.add(student1);
      students.add(student2);
   }
   @Override
   public void deleteStudent(Student student) {
      students.remove(student.getRollNo());
      System.out.println("Student: Roll No " + student.getRollNo()
         +", deleted from database");
   }

   //从数据库中检索学生名单
   @Override
   public List<Student> getAllStudents() {
      return students;
   }

   @Override
   public Student getStudent(int rollNo) {
      return students.get(rollNo);
   }

   @Override
   public void updateStudent(Student student) {
      students.get(student.getRollNo()).setName(student.getName());
      System.out.println("Student: Roll No " + student.getRollNo()
         +", updated in the database");
   }
}
```

步骤 4

使用 *StudentDao* 来演示数据访问对象模式的用法。

CompositeEntityPatternDemo.java

```
public class DaoPatternDemo {
   public static void main(String[] args) {
      StudentDao studentDao = new StudentDaoImpl();

      //输出所有的学生
      for (Student student : studentDao.getAllStudents()) {
         System.out.println("Student: [RollNo : "
            +student.getRollNo()+", Name : "+student.getName()+" ]");
      }

      //更新学生
      Student student =studentDao.getAllStudents().get(0);
      student.setName("Michael");
      studentDao.updateStudent(student);

      //获取学生
      studentDao.getStudent(0);
      System.out.println("Student: [RollNo : "
         +student.getRollNo()+", Name : "+student.getName()+" ]");
   }
}
```

步骤 5

验证输出。

```
Student: [RollNo : 0, Name : Robert ]
Student: [RollNo : 1, Name : John ]
Student: Roll No 0, updated in the database
Student: [RollNo : 0, Name : Michael ]
```

前端控制器模式

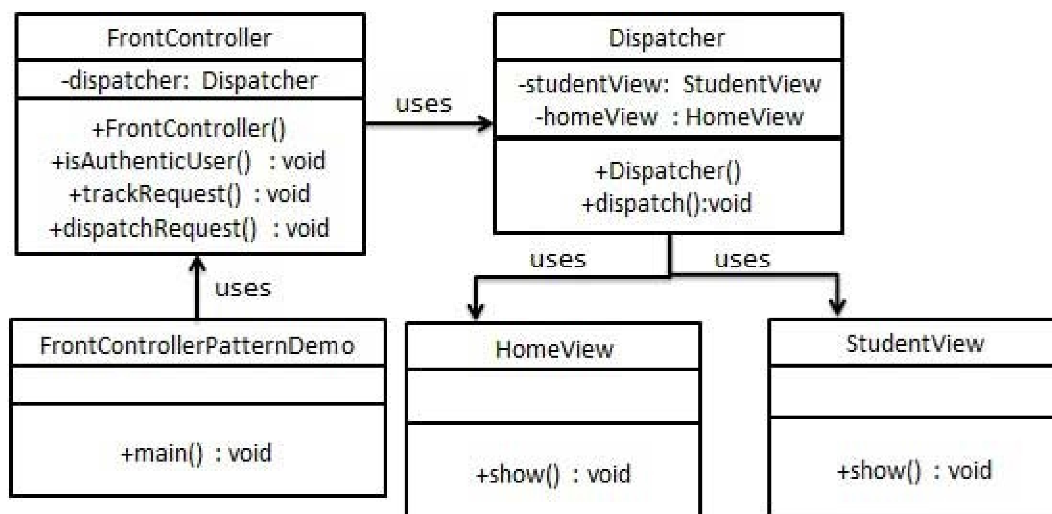
前端控制器模式（Front Controller Pattern）是用来提供一个集中的请求处理机制，所有的请求都将由一个单一的处理程序处理。该处理程序可以做认证/授权/记录日志，或者跟踪请求，然后把请求传给相应的处理程序。以下是这种设计模式的实体。

- 前端控制器（**Front Controller**） - 处理应用程序所有类型请求的单个处理程序，应用程序可以是基于 web 的应用程序，也可以是基于桌面的应用程序。
- 调度器（**Dispatcher**） - 前端控制器可能使用一个调度器对象来调度请求到相应的具体处理程序。
- 视图（**View**） - 视图是为请求而创建的对象。

实现

我们将创建 *FrontController*、*Dispatcher* 分别当作前端控制器和调度器。*HomeView* 和 *StudentView* 表示各种为前端控制器接收到的请求而创建的视图。

FrontControllerPatternDemo，我们的演示类使用 *FrontController* 来演示前端控制器设计模式。



步骤 1

创建视图。

HomeView.java

```
public class HomeView {
   public void show(){
      System.out.println("Displaying Home Page");
   }
}
```

StudentView.java

```
public class StudentView {
   public void show(){
      System.out.println("Displaying Student Page");
   }
}
```

步骤 2

创建调度器 Dispatcher。

Dispatcher.java

```
public class Dispatcher {
   private StudentView studentView;
   private HomeView homeView;
   public Dispatcher(){
      studentView = new StudentView();
      homeView = new HomeView();
   }

   public void dispatch(String request){
      if(request.equalsIgnoreCase("STUDENT")){
         studentView.show();
      }else{
         homeView.show();
      }
   }
}
```

步骤 3

创建前端控制器 FrontController。

Context.java

```
public class FrontController {

   private Dispatcher dispatcher;

   public FrontController(){
      dispatcher = new Dispatcher();
   }

   private boolean isAuthenticUser(){
      System.out.println("User is authenticated successfully.");
      return true;
   }

   private void trackRequest(String request){
      System.out.println("Page requested: " + request);
   }

   public void dispatchRequest(String request){
      //记录每一个请求
      trackRequest(request);
      //对用户进行身份验证
      if(isAuthenticUser()){
         dispatcher.dispatch(request);
      }
   }
}
```

步骤 4

使用 *FrontController* 来演示前端控制器设计模式。

FrontControllerPatternDemo.java

```
public class FrontControllerPatternDemo {
   public static void main(String[] args) {
      FrontController frontController = new FrontController();
      frontController.dispatchRequest("HOME");
      frontController.dispatchRequest("STUDENT");
   }
}
```

步骤 5

验证输出。

```
Page requested: HOME
User is authenticated successfully.
Displaying Home Page
Page requested: STUDENT
User is authenticated successfully.
Displaying Student Page
```

拦截过滤器模式

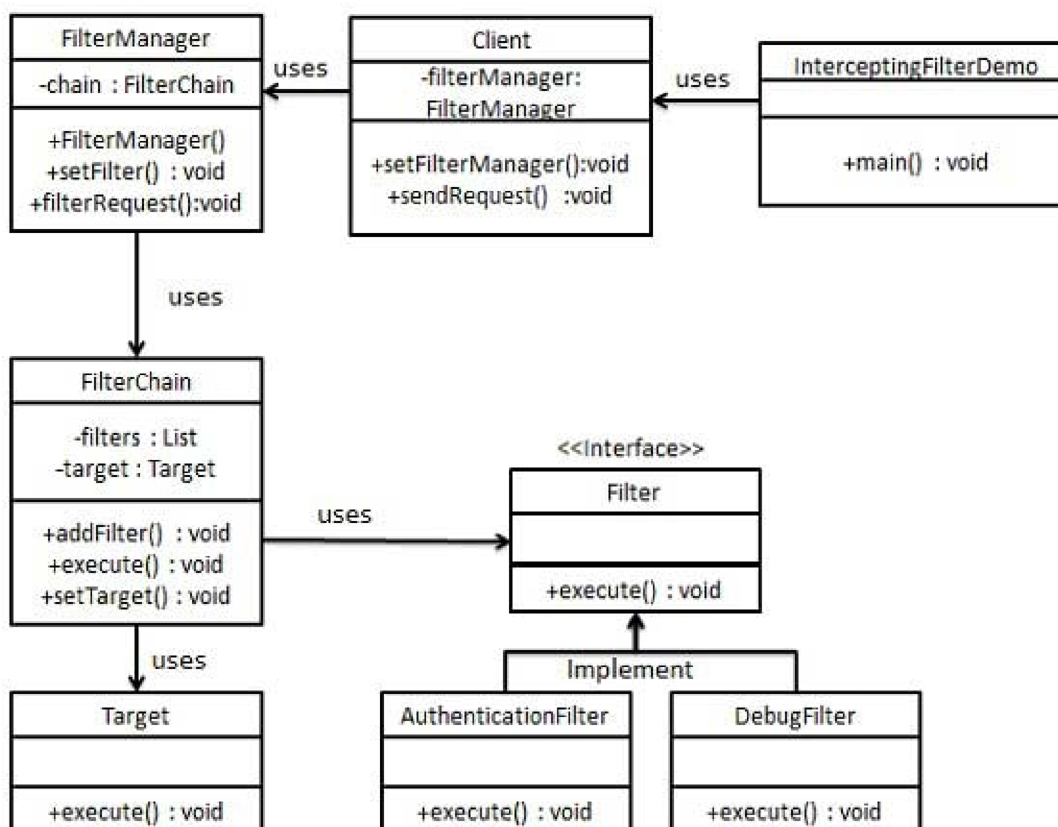
拦截过滤器模式（Intercepting Filter Pattern）用于对应用程序的请求或响应做一些预处理/后处理。定义过滤器，并在把请求传给实际目标应用程序之前应用在请求上。过滤器可以做认证/授权/记录日志，或者跟踪请求，然后把请求传给相应的处理程序。以下是这种设计模式的实体。

- 过滤器（**Filter**） - 过滤器在请求处理程序执行请求之前或之后，执行某些任务。
- 过滤器链（**Filter Chain**） - 过滤器链带有多个过滤器，并在 Target 上按照定义的顺序执行这些过滤器。
- **Target** - Target 对象是请求处理程序。
- 过滤管理器（**Filter Manager**） - 过滤管理器管理过滤器和过滤器链。
- 客户端（**Client**） - Client 是向 Target 对象发送请求的对象。

实现

我们将创建 *FilterChain*、*FilterManager*、*Target*、*Client* 作为表示实体的各种对象。*AuthenticationFilter* 和 *DebugFilter* 表示实体过滤器。

InterceptingFilterDemo，我们的演示类使用 *Client* 来演示拦截过滤器设计模式。



步骤 1

创建过滤器接口 Filter。

Filter.java

```
public interface Filter {
   public void execute(String request);
}
```

步骤 2

创建实体过滤器。

AuthenticationFilter.java

```
public class AuthenticationFilter implements Filter {
   public void execute(String request){
      System.out.println("Authenticating request: " + request);
   }
}
```

DebugFilter.java

```
public class DebugFilter implements Filter {
   public void execute(String request){
      System.out.println("request log: " + request);
   }
}
```

步骤 3

创建 Target。

Target.java

```
public class Target {
   public void execute(String request){
      System.out.println("Executing request: " + request);
   }
}
```

步骤 4

创建过滤器链。

FilterChain.java

```
import java.util.ArrayList;
import java.util.List;

public class FilterChain {
   private List<Filter> filters = new ArrayList<Filter>();
   private Target target;

   public void addFilter(Filter filter){
      filters.add(filter);
   }

   public void execute(String request){
      for (Filter filter : filters) {
         filter.execute(request);
      }
      target.execute(request);
   }

   public void setTarget(Target target){
      this.target = target;
   }
}
```

步骤 5

创建过滤管理器。

FilterManager.java

```
public class FilterManager {
   FilterChain filterChain;

   public FilterManager(Target target){
      filterChain = new FilterChain();
      filterChain.setTarget(target);
   }
   public void setFilter(Filter filter){
      filterChain.addFilter(filter);
   }

   public void filterRequest(String request){
      filterChain.execute(request);
   }
}
```

步骤 6

创建客户端 Client。

Client.java

```
public class Client {
   FilterManager filterManager;

   public void setFilterManager(FilterManager filterManager){
      this.filterManager = filterManager;
   }

   public void sendRequest(String request){
      filterManager.filterRequest(request);
   }
}
```

步骤 7

使用 *Client* 来演示拦截过滤器设计模式。

FrontControllerPatternDemo.java

```
public class InterceptingFilterDemo {
   public static void main(String[] args) {
      FilterManager filterManager = new FilterManager(new Target());
      filterManager.setFilter(new AuthenticationFilter());
      filterManager.setFilter(new DebugFilter());

      Client client = new Client();
      client.setFilterManager(filterManager);
      client.sendRequest("HOME");
   }
}
```

步骤 8

验证输出。

```
Authenticating request: HOME
request log: HOME
Executing request: HOME
```

服务定位器模式

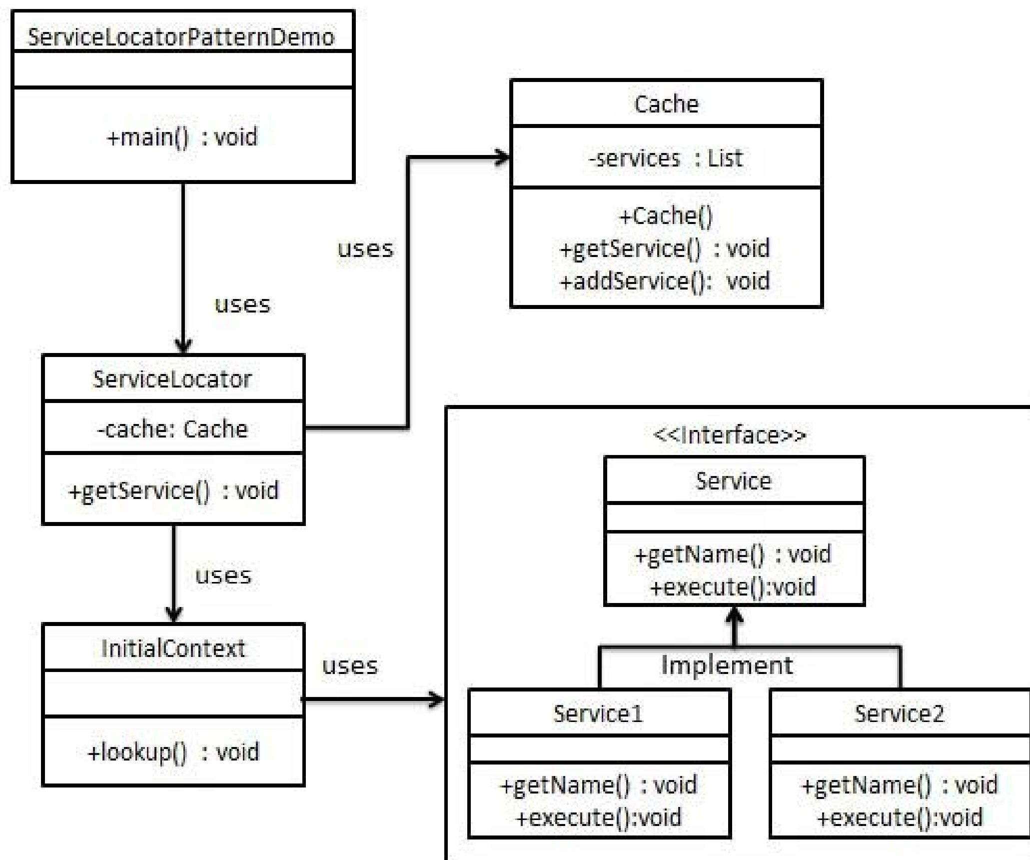
服务定位器模式（Service Locator Pattern）用在我们想使用 JNDI 查询定位各种服务的时候。考虑到为某个服务查找 JNDI 的代价很高，服务定位器模式充分利用了缓存技术。在首次请求某个服务时，服务定位器在 JNDI 中查找服务，并缓存该服务对象。当再次请求相同的服务时，服务定位器会在它的缓存中查找，这样可以在很大程度上提高应用程序的性能。以下是这种设计模式的实体。

- 服务（**Service**） - 实际处理请求的服务。对这种服务的引用可以在 JNDI 服务器中查找到。
- **Context /** 初始的 **Context** - JNDI Context 带有对要查找的服务的引用。
- 服务定位器（**Service Locator**） - 服务定位器是通过 JNDI 查找和缓存服务来获取服务的单点接触。
- 缓存（**Cache**） - 缓存存储服务的引用，以便复用它们。
- 客户端（**Client**） - Client 是通过 ServiceLocator 调用服务的对象。

实现

我们将创建 *ServiceLocator*、*InitialContext*、*Cache*、*Service* 作为表示实体的各种对象。*Service1* 和 *Service2* 表示实体服务。

ServiceLocatorPatternDemo，我们的演示类在这里是作为一个客户端，将使用 *ServiceLocator* 来演示服务定位器设计模式。



步骤 1

创建服务接口 Service。

Service.java

```
public interface Service {
   public String getName();
   public void execute();
}
```

步骤 2

创建实体服务。

Service1.java

```
public class Service1 implements Service {
   public void execute(){
      System.out.println("Executing Service1");
   }

   @Override
   public String getName() {
      return "Service1";
   }
}
```

Service2.java

```
public class Service2 implements Service {
   public void execute(){
      System.out.println("Executing Service2");
   }

   @Override
   public String getName() {
      return "Service2";
   }
}
```

步骤 3

为 JNDI 查询创建 InitialContext。

InitialContext.java

```
public class InitialContext {
   public Object lookup(String jndiName){
      if(jndiName.equalsIgnoreCase("SERVICE1")){
         System.out.println("Looking up and creating a new Service1 object");
         return new Service1();
      }else if (jndiName.equalsIgnoreCase("SERVICE2")){
         System.out.println("Looking up and creating a new Service2 object");
         return new Service2();
      }
      return null;
   }
}
```

步骤 4

创建缓存 Cache。

Cache.java

```
import java.util.ArrayList;
import java.util.List;

public class Cache {

   private List<Service> services;

   public Cache(){
      services = new ArrayList<Service>();
   }

   public Service getService(String serviceName){
      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(serviceName)){
            System.out.println("Returning cached  "+serviceName+" object");
            return service;
         }
      }
      return null;
   }

   public void addService(Service newService){
      boolean exists = false;
      for (Service service : services) {
         if(service.getName().equalsIgnoreCase(newService.getName())){
            exists = true;
         }
      }
      if(!exists){
         services.add(newService);
      }
   }
}
```

步骤 5

创建服务定位器。

ServiceLocator.java

```
public class ServiceLocator {
   private static Cache cache;

   static {
      cache = new Cache();
   }

   public static Service getService(String jndiName){

      Service service = cache.getService(jndiName);

      if(service != null){
         return service;
      }

      InitialContext context = new InitialContext();
      Service service1 = (Service)context.lookup(jndiName);
      cache.addService(service1);
      return service1;
   }
}
```

步骤 6

使用 *ServiceLocator* 来演示服务定位器设计模式。

ServiceLocatorPatternDemo.java

```
public class ServiceLocatorPatternDemo {
   public static void main(String[] args) {
      Service service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
      service = ServiceLocator.getService("Service1");
      service.execute();
      service = ServiceLocator.getService("Service2");
      service.execute();
   }
}
```

步骤 7

验证输出。

```
Looking up and creating a new Service1 object
Executing Service1
Looking up and creating a new Service2 object
Executing Service2
Returning cached  Service1 object
Executing Service1
Returning cached  Service2 object
Executing Service2
```

传输对象模式

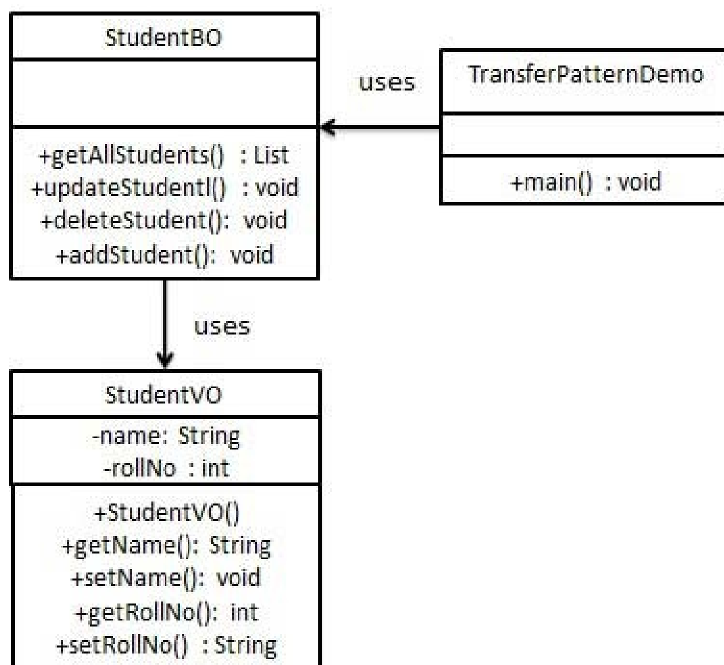
传输对象模式（Transfer Object Pattern）用于从客户端向服务器一次性传递带有多个属性的数据。传输对象也被称为数值对象。传输对象是一个具有 getter/setter 方法的简单的 POJO 类，它是可序列化的，所以它可以通过网络传输。它没有任何的行为。服务器端的业务类通常从数据库读取数据，然后填充 POJO，并把它发送到客户端或按值传递它。对于客户端，传输对象是只读的。客户端可以创建自己的传输对象，并把它传递给服务器，以便一次性更新数据库中的数值。以下是这种设计模式的实体。

- 业务对象（**Business Object**） - 为传输对象填充数据的业务服务。
- 传输对象（**Transfer Object**） - 简单的 POJO，只有设置/获取属性的方法。
- 客户端（**Client**） - 客户端可以发送请求或者发送传输对象到业务对象。

实现

我们将创建一个作为业务对象的 *StudentBO* 和作为传输对象的 *StudentVO*，它们都代表了我们的实体。

TransferObjectPatternDemo，我们的演示类在这里是作为一个客户端，将使用 *StudentBO* 和 *Student* 来演示传输对象设计模式。



步骤 1

创建传输对象。

StudentVO.java

```
public class StudentVO {
   private String name;
   private int rollNo;

   StudentVO(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getRollNo() {
      return rollNo;
   }

   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
}
```

步骤 2

创建业务对象。

StudentBO.java

```
import java.util.ArrayList;
import java.util.List;

public class StudentBO {

   //列表是当作一个数据库
   List<StudentVO> students;

   public StudentBO(){
      students = new ArrayList<StudentVO>();
      StudentVO student1 = new StudentVO("Robert",0);
      StudentVO student2 = new StudentVO("John",1);
      students.add(student1);
      students.add(student2);
   }
   public void deleteStudent(StudentVO student) {
      students.remove(student.getRollNo());
      System.out.println("Student: Roll No "
      + student.getRollNo() +", deleted from database");
   }

   //从数据库中检索学生名单
   public List<StudentVO> getAllStudents() {
      return students;
   }

   public StudentVO getStudent(int rollNo) {
      return students.get(rollNo);
   }

   public void updateStudent(StudentVO student) {
      students.get(student.getRollNo()).setName(student.getName());
      System.out.println("Student: Roll No "
      + student.getRollNo() +", updated in the database");
   }
}
```

步骤 3

使用 *StudentBO* 来演示传输对象设计模式。

TransferObjectPatternDemo.java

```
public class TransferObjectPatternDemo {
   public static void main(String[] args) {
      StudentBO studentBusinessObject = new StudentBO();

      //输出所有的学生
      for (StudentVO student : studentBusinessObject.getAllStudents()) {
         System.out.println("Student: [RollNo : "
         +student.getRollNo()+", Name : "+student.getName()+" ]");
      }

      //更新学生
      StudentVO student =studentBusinessObject.getAllStudents().get(0);
      student.setName("Michael");
      studentBusinessObject.updateStudent(student);

      //获取学生
      studentBusinessObject.getStudent(0);
      System.out.println("Student: [RollNo : "
      +student.getRollNo()+", Name : "+student.getName()+" ]");
   }
}
```

步骤 4

验证输出。

```
Student: [RollNo : 0, Name : Robert ]
Student: [RollNo : 1, Name : John ]
Student: Roll No 0, updated in the database
Student: [RollNo : 0, Name : Michael ]
```

Git教程

Git 诞生于一个极富纷争大举创新的年代。Linux 内核开源项目有着为数众广的参与者。绝大多数的 Linux 内核维护工作都花在了提交补丁和保存归档的繁琐事务上（1991－2002年间）。到 2002 年，整个项目组开始启用分布式版本控制系统 BitKeeper 来管理和维护代码。

到了 2005 年，开发 BitKeeper 的商业公司同 Linux 内核开源社区的合作关系结束，他们收回了免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux 的缔造者 Linus Torvalds ）不得不吸取教训，只有开发一套属于自己的版本控制系统才不至于重蹈覆辙。他们对新的系统制订了若干目标：

- 速度
- 简单的设计
- 对非线性开发模式的强力支持（允许上千个并行开发的分支）
- 完全分布式
- 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）

自诞生于 2005 年以来，Git 日臻成熟完善，在高度易用的同时，仍然保留着初期设定的目标。它的速度飞快，极其适合管理大项目，它还有着令人难以置信的非线性分支管理系统（见第三章），可以应付各种复杂的项目开发需求。

Git 教程

Git 是一个分布式的版本控制和源代码管理系统，强调速度。 Git 最初由Linus Torvalds设计和开发为Linux内核开发管理代码。 Git是GNU通用公共许可证版本2的条款下分发的免费软件。

本教程将教你如何使用Git 在你的项目版本控制在分布式环境中的基于 Web 和非基于Web 应用程序的开发工作。

读者

对于初学者来说已经准备本教程，帮助他们了解Git版本控制系统的基本功能。完成本教程后，可以把帮助你熟悉和使用Git版本控制系统。

前提条件

我们假设你要使用 Git 来处理各级 Java 和非Java项目。所以如果你有知识，开发的基于 Web 和非基于 Web 的应用程序的软件开发生命周期和知识，将有助于学习和使用Git。

Git 基本概念 - Git教程

版本控制系统 (VCS)

版本控制系统 (VCS) 是软件，帮助软件开发人员携手合作，他们的工作并保持完整的历史。

以下是VCS目标

1. 允许开发人员同步工作.
2. 不要覆盖对方的变化.
3. 维护历史的每一个版本.

以下是常见的VCS

1. 集中式版本控制系统（CVCS）
2. 分散式/分布式版本控制系统（DVCS）

在这个教程，我们将介绍集中分布式的版本控制系统，尤其是Git。 Git 属于分布式版本控制系统。

分布式版本控制系统（DVCS）

集中式版本控制系统采用中央服务器上存储的所有文件和实现团队协作。但是CVCS主要缺点是中央服务器的单点故障，即故障。不幸的是，如果中央服务器宕机一小时，然后在该时段没有人可以合作。即使在最坏的情况下，如果中央服务器的磁盘被损坏，并没有采取适当的备份，那么将失去整个项目的历史。

DVCS客户不仅检出的最新快照目录，但他们也完全反映资源库。如果SEVER停机，然后从任何客户端库可以复制回服务器，以恢复它。每个结账是完整的版本库备份。 Git不会依赖中央服务器，这就是为什么可以执行许多操作，当处于脱机状态。可以提交修改，创建分支视图日志和执行其他操作，当处于脱机状态。只需要网络连接，发布您的更改，并采用最新变化。

Git优势

Git是GPL开源许可证下发布的。它可自由在互联网上。可以使用Git来管理项目无需支付一分钱。由于它是开源的，可以下载它的源代码，并根据要求进行更改。

由于大部分的操作都在本地执行，它给人带来巨大的好处，在速度方面。 Git 不依赖于中央服务器，为什么每一个操作就没有必要与远程服务器进行交互。 Git核心部分是写在C中，从而避免了与其他高级语言的运行时开销。虽然 Git中反映整个存储库，在客户端上的数据的大小是小的。这说明它是在客户端上的数据压缩和存储的效率有多高。

丢失数据的机会是非常罕见的，当有多个副本。存在于任何客户端的数据存储库中，因此它可以被用来在发生崩溃或磁盘损坏的镜像。

Git使用常见的加密散列函数，称为安全的哈希算法（SHA1）命名，并确定在其数据库中的对象。每一个文件并提交检查总结和检索其校验在结账的时候。意思是说，这是不可能改变文件，日期，提交信息和且从Git 数据库不知道Git 任何其他数据。

在CVCS情况下，中心服务器需要足够强大，要求整个团队服务。对于较小的团队，这是不是一个问题，但随着团队规模的增长，服务器的硬件限制可能成为一个性能瓶颈。在DVCS开发的情况下不与服务器进行交互，除非他们需要推或拉的变化。所有繁重发生在客户端上，所以服务器硬件可以是很简单的。

CVCS使用廉价的复制机制，这意味着如果我们创建新的分支，它会复制到新分支的所有代码，所以它的耗时和效率不高。另外CVCS的分支的删除和合并是复杂和费时的。但是，使用Git分支管理是很简单的。只需要几秒钟，创建，删除和合并分支。

1. 自由和开放源码
2. 快速和小
3. 隐式备份
4. 安全
5. 不需要强大的硬件
6. 更简单的分支

DVCS术语

本地资源库

每个VCS工具提供私有工作场所的工作副本。开发者在他的私人工作场所的变化，并提交这些更改后成为仓库的一部分。 Git的需要这一步，为他们提供的专用副本是整个仓库。用户可以执行许多操作，这个库中，如添加文件，删除文件，重命名文件，移动文件，提交改变，还有更多。

工作目录，暂存区域或索引

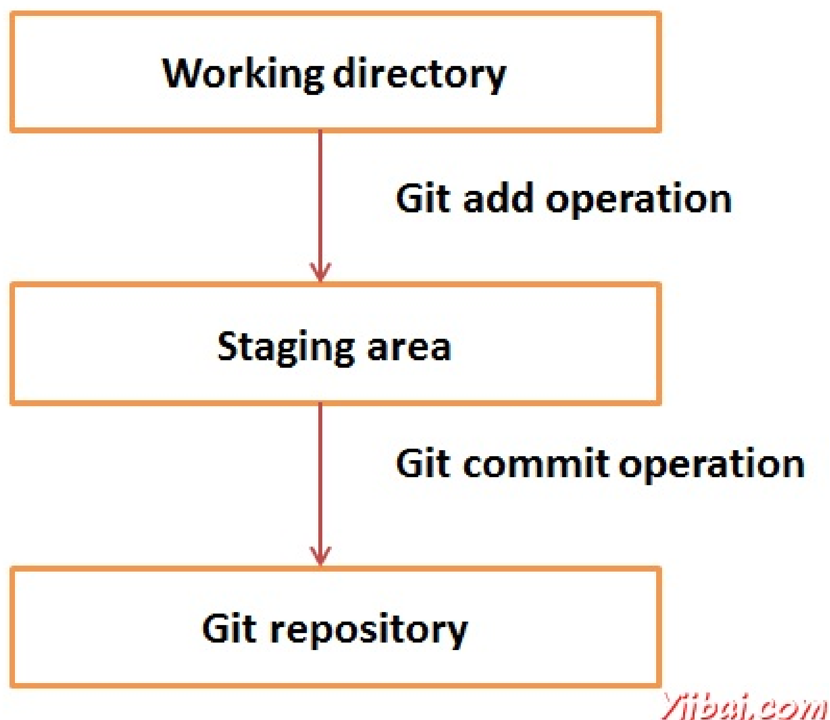
工作目录是地方文件检出。其他CVCS开发商一般不修改，并承诺他的变化，直接向版本库。但Git使用不同的策略。 Git不会跟踪每一个修改过的文件。每当提交操作，Git在目前临时区域的文件。只有文件被认为是目前在临时区域提交，而不是所有修改过的文件。

让我们来看看Git的基本工作流程。

第1步：修改文件的工作目录。

第2步：将这些文件添加到暂存区

第3步：执行commit操作。这将文件从临时区域。推送操作后，它永久地存储更改的Git仓库



假设修改了两个文件，即 “sort.c” and “search.c”， 两种不同分别 提交操作。可以添加一个文件分段区域，不提交。第一次提交后重复相同的步骤为另一个文件。

```
# First commit
[bash]$ git add sort.c

# adds file to the staging area
[bash]$ git commit –m “Added sort operation”
```

```
# Second commit
[bash]$ git add search.c

# adds file to the staging area
[bash]$ git commit –m “Added search operation”
```

BLOBS

BLOB代表二进制大对象。为代表 blob文件的每个版本。一个blob保存文件数据，但不包含任何有关文件的元数据。它是一个二进制文件，该文件它被命名为SHA1哈希 Git数据库中。在Git中，文件未提及的名字。一切固定内容寻址。

Tree

树是一个对象，它表示一个目录。它拥有blobs以及其他子目录。一棵树是一个二进制文件，该文件存储Blob树，也被命名为树对象的SHA1哈希的引用。

Commit

提交持有的库的当前状态。COMMIT命令同样由SHA1哈希的名字命名。可以考虑commit对象的链表节点。每个提交的对象有父commit 对象的指针。从给定的承诺可以遍历寻找在父指针，查看历史记录的提交。如果提交多个父承诺，这意味着特定的提交是由两个分支合并。

BRANCHES

分支用来创建另一条线的发展。默认情况下，Git的主分支，这是一样躯干颠覆。平时要工作的新功能创建一个分支。功能完成后，它被合并回master分支，我们删除分支。每个分支所引用HEAD，这点在分支的最新提交。每当做出了一个提交，HEAD更新为最新提交。

TAGS

包括特定版本库中的标签分配一个有意义的名称。标签是非常相似的分支，但不同的是，标签是不可改变的。手段标记的一个分支，没有人打算修改。一旦标签被创建为特定的提交，即使创建一个新的提交，也不会被更新。通常开发人员创建标签的产品发布。

CLONE

克隆操作的库创建实例。克隆操作不仅检出的工作拷贝，但它也反映了完整的信息库。用户可以执行许多操作，这个本地仓库。网络介入是唯一的一次，当正在同步资料库实例。

PULL

Pull操作复制的变化，本地的一个实例来从远程仓库。Pull操作是用于两个存储库实例之间的同步。这是在Subversion更新操作一样。

PUSH

推动从本地存储库实例的远程操作副本的变化。这是用来储存到Git仓库中永久改变。这是在Subversion的提交操作相同。

HEAD

HEAD指针总是指向分支的最新提交。每当你做出了一个提交，HEAD更新为最新提交。HEAD树枝存储在.git/refs/heads/ 目录中。

```
[CentOS]$ ls -1 .git/refs/heads/
master

[CentOS]$ cat .git/refs/heads/master
570837e7d58fa4bccd86cb575d884502188b0c49
```

REVISION

修订版本的源代码。在Git修订代表的提交。这些提交由SHA1安全哈希值确定。

URL

URL代表的Git仓库的位置。 Git 的URL存储在配置文件中。

```
[tom@CentOS tom_repo]$ pwd
/home/tom/tom_repo

[tom@CentOS tom_repo]$ cat .git/config
[core]
repositoryformatversion = 0
filemode = true
bare = false
logallrefupdates = true
[remote "origin"]
**url = gituser@git.server.com:project.git**
fetch = +refs/heads/*:refs/remotes/origin/*
```

Git 环境设置（安装） - Git教程

在使用Git之前，必须安装它，并做一些基本配置的变化。下面是步骤在Ubuntu和CentOS Linux安装 Git 客户端。

Git客户端安装

如果使用的是GNU/ Linux 发行版Debian基本apt-get命令就可以搞定一切。

```
[ubuntu ~]$ sudo apt-get install git-core
[sudo] password for ubuntu:

[ubuntu ~]$ git --version
git version 1.8.1.2
```

而且，如果使用的是基于RPM的GNU/ Linux发行版使用yum命令，如下：

```
[CentOS ~]$
su -
Password:

[CentOS ~]# yum -y install git-core

[CentOS ~]# git --version
git version 1.7.1
```

自定义Git环境

Git提供git 的配置工具，它允许设置配置变量。 Git会把所有的全局配置.gitconfig 文件位于主目录。要设置这些为全局配置值，添加 -global选项，如果省略 -global选项，那么配置是具体当前的Git存储库。

还可以设置系统范围内的配置。 Git存储这些值是在/etc/gitconfig文件，其中包含的配置系统上的每一个用户和资源库。要设置这些值，必须有root权限，并使用 -system 选项。

上面的代码编译和执行时，它会产生以下结果：

设置用户名

此信息用于Git的每个提交。

```
[jerry@CentOS project]$ git config --global user.name "Jerry Mouse"
```

设置电子邮件ID

此信息用于Git的每个提交。

```
[jerry@CentOS project]$ git config --global user.email "jerry@yiibai.com"
```

避免PULLING提交合并

先从远程资源库的最新变化，如果这些变化是不同的，默认情况下，Git 创建合并提交。我们可以通过以下设置来避免这种。

```
jerry@CentOS project]$ git config --global branch.autosetuprebase always
```

颜色高亮

下面的命令使颜色突出显示在控制台的Git。

```
[jerry@CentOS project]$ git config --global color.ui true

[jerry@CentOS project]$ git config --global color.status auto

[jerry@CentOS project]$ git config --global color.branch auto
```

设置默认编辑器

默认情况下，Git的使用系统默认取自VISUAL或EDITOR环境变量的编辑器。我们可以设定一个不同的使用git 配置。

```
[jerry@CentOS project]$ git config --global core.editor vim
```

设置默认的合并工具

Git不会提供一个默认的合并工具整合到工作树冲突的更改。我们可以设置默认的合并工具，通过启用以下设置。

```
[jerry@CentOS project]$ git config --global merge.tool vimdiff
```

列出GIT设置

为了验证自己的Git设置本地存储库使用git 的config-list命令，如下所示。

```
[jerry@CentOS ~]$ git config --list
```

上面的命令会产生以下结果。

```
user.name=Jerry Mouse
user.email=jerry@yiibai.com
push.default=nothing
branch.autosetuprebase=always
color.ui=true
color.status=auto
color.branch=auto
core.editor=vim
merge.tool=vimdiff
```

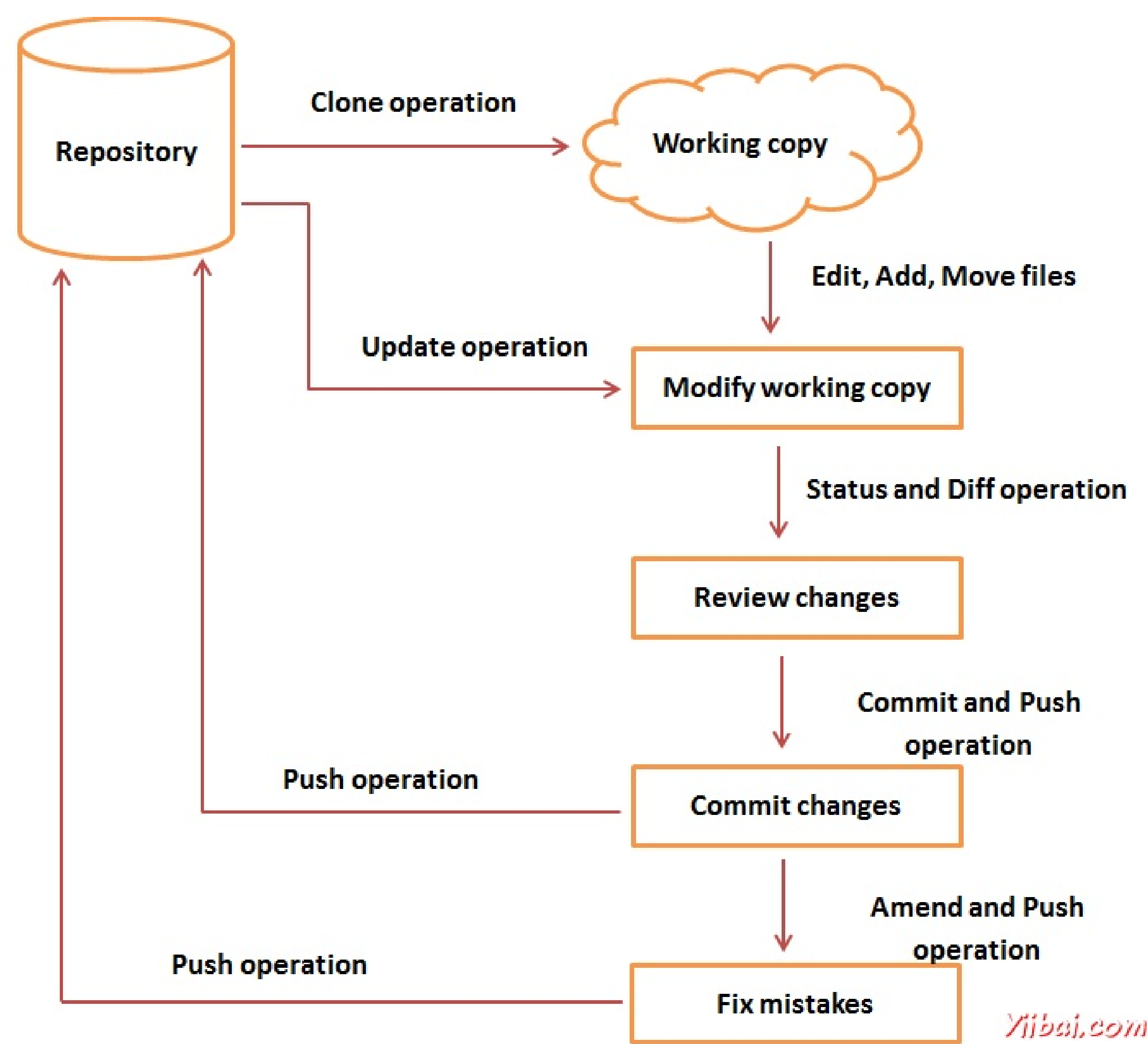
Git 生命周期 - Git教程

在本章中，我们将讨论的Git的生命周期。在后面的章节中，我们将看到的Git命令为每个操作。

一般工作流程是这样的：

1. 克隆Git仓库作为工作副本。
2. 可以添加/编辑文件，修改工作副本。
3. 如果有必要，你还服用其他开发人员的变化，更新工作副本。
4. 审查前提交。
5. 提交修改。如果一切都很好，然后推到存储库的更改。
6. 提交之后，如果知道是什么错误，那么纠正最后一次提交，并推送修改到版本库。

以下是工作流程的图形表示。


Repository
Clone operation
Working copy
Edit, Add, Move files
Update operation
Modify working copy
Status and Diff operation
Review changes
Commit and Push operation
Push operation
Commit changes
Amend and Push operation
Push operation
Fix mistakes
Yiibai.com

Git 创建操作 - Git教程

在本章中，我们将看到如何创建一个远程Git仓库，从现在开始，我们将会把它作为Git服务器。我们需要一个的Git服务器允许团队协作。

创建新用户

```
# add new group
[root@CentOS ~]# groupadd dev

# add new user
[root@CentOS ~]# useradd -G devs -d /home/gituser -m -s /bin/bash gituser

# change password
[root@CentOS ~]# passwd gituser
```

上面的命令会产生以下结果。

```
Changing password for user gituser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

创建一个裸库

让我们初始化一个新的资料库使用init命令后面加上 -bare选项。它初始化没有工作目录库。按照惯例裸库必须命名为 .git。

```
[gituser@CentOS ~]$ pwd
/home/gituser

[gituser@CentOS ~]$ mkdir project.git

[gituser@CentOS ~]$ cd project.git/

[gituser@CentOS project.git]$ ls

[gituser@CentOS project.git]$ git --bare init
Initialized empty Git repository in /home/gituser-m/project.git/

[gituser@CentOS project.git]$ ls
branches config description HEAD hooks info objects refs
```

生成公共/私有RSA密钥对

让我们遍历Git服务器端的配置过程中，使用ssh-keygen实用程序生成公共/私有RSA密钥对，我们将使用这些键进行用户认证。

打开一个终端并输入以下命令，直接按回车为每个输入。成功完成后，它会创建主目录 .ssh 目录内。

```
tom@CentOS ~]$ pwd
/home/tom

[tom@CentOS ~]$ ssh-keygen
```

上面的命令会产生以下结果。

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/tom/.ssh/id_rsa): **Press Enter Only**
Created directory '/home/tom/.ssh'.
Enter passphrase (empty for no passphrase): **---------------> Press Enter Only**
Enter same passphrase again: **------------------------------> Press Enter Only**
Your identification has been saved in /home/tom/.ssh/id_rsa.
Your public key has been saved in /home/tom/.ssh/id_rsa.pub.
The key fingerprint is:
df:93:8c:a1:b8:b7:67:69:3a:1f:65:e8:0e:e9:25:a1 tom@CentOS
The key's randomart image is:
+--[ RSA 2048]----+
| |
| |
| |
|
.
|
| Soo |
| o*B. |
| E = *.= |
| oo==. . |
| ..+Oo
|
+-----------------+
```

ssh-keygen 已经产生了两个键，第一个是私有的（即id_rsa），另一个是公共（即id_rsa.pub 文件）。

> 注: 切勿与他人共享你的私钥。

添加键 authorized_keys

假设有两个开发项目即Tom 和Jerry工作。两个用户生成公钥。让我们来看看如何使用这些密钥进行身份验证。

Tom 添加他的公钥服务器使用 ssh-copy-id这个命令下面给出

```
[tom@CentOS ~]$ pwd
/home/tom

[tom@CentOS ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub gituser@git.server.com
```

上面的命令会产生以下结果。

```
gituser@git.server.com's password:
Now try logging into the machine, with "ssh 'gituser@git.server.com'", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

同样，Jerry 也增加了他的公共密钥服务器使用 ssh-copy-id 这个命令。

```
[jerry@CentOS ~]$ pwd
/home/jerry

[jerry@CentOS ~]$ ssh-copy-id -i ~/.ssh/id_rsa gituser@git.server.com
```

上面的命令会产生以下结果。

```
gituser@git.server.com's password:
Now try logging into the machine, with "ssh 'gituser@git.server.com'", and check in:
.ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

推修改到版本库

我们已经创建了裸库在服务器上，并允许两个用户访问。从现在Tom 和 Jerry 可以把他们修改到版本库，将其添加为远程。

Git的init命令创建 .git 目录来存储元数据的存储库。每次读取配置从 .git/config 文件.

Tom 创建一个新的目录，添加READE文件作为初始提交并提交他的变化。提交后，他确认提交信息，运行git日志命令。

```
[tom@CentOS ~]$ pwd
/home/tom

[tom@CentOS ~]$ mkdir tom_repo

[tom@CentOS ~]$ cd tom_repo/

[tom@CentOS tom_repo]$ git init
Initialized empty Git repository in /home/tom/tom_repo/.git/

[tom@CentOS tom_repo]$ echo 'TODO: Add contents for README' > README

[tom@CentOS tom_repo]$ git status -s
?? README

[tom@CentOS tom_repo]$ git add .

[tom@CentOS tom_repo]$ git status -s
A README

[tom@CentOS tom_repo]$ git commit -m 'Initial commit'
```

上面的命令会产生以下结果。

```
[master (root-commit) 19ae206] Initial commit
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 README
```

Tom 执行git 的日志命令，检查日志消息。

```
[tom@CentOS tom_repo]$ git log
```

上面的命令会产生以下结果。

```
commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit</tom@yiibai.com>
```

Tom 提交了他的变化到本地资源库。现在是时候将更改到远程仓库。但在此之前，我们必须添加作为远程仓库，这是一个时间的操作。在此之后，他可以放心地推送到远程存储库的更改。

> 注: 默认情况下，Git的推到匹配的分支：对于每一个分支退出本地端的远程端更新，如果已经存在具有相同名称的一个分支。在我们的教程每次我推原点主分支的变化，根据您的要求，使用适当的分支名。

```
[tom@CentOS tom_repo]$ git remote add origin gituser@git.server.com:project.git

[tom@CentOS tom_repo]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 3, done.
Writing objects: 100% (3/3), 242 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
* [new branch]
master −> master
```

现在更改成功提交到远程仓库。

Git 克隆操作 - Git教程

我们有一个裸库Git服务器，Tom 也推了他的第一个版本。现在，Jerry 可以查看他的变化。克隆操作的远程存储库创建实例。

Jerry 在他的home目录，并创建新的目录，执行克隆操作。

```
[jerry@CentOS ~]$ mkdir jerry_repo

[jerry@CentOS ~]$ cd jerry_repo/

[jerry@CentOS jerry_repo]$ git clone gituser@git.server.com:project.git
```

上面的命令会产生以下结果。

```
Initialized empty Git repository in /home/jerry/jerry_repo/project/.git/
remote: Counting objects: 3, done.
Receiving objects: 100% (3/3), 241 bytes, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
```

Jerry 改变目录到新的本地存储库，并列出目录内容。

```
[jerry@CentOS jerry_repo]$ cd project/

[jerry@CentOS jerry_repo]$ ls
README
```

Git 执行更改 - Git教程

Jerry 克隆库，他决定实现基本字符串操作。于是，他创建文件string.c，在添加内容到string.c 会这个样子。

```
#include <stdio.h>

int my_strlen(char *s)
{
   char *p = s;

   while (*p)
      ++p;

   return (p - s);
}

int main(void)
{
   int i;
   char *s[] = {
      "Git tutorials",
      "Tutorials Yiibai"
   };

   for (i = 0; i < 2; ++i)
      printf("string lenght of %s = %d
", s[i], my_strlen(s[i]));

   return 0;
}
```

他编译和测试代码，一切工作正常。现在，他可以放心地添加这些修改到版本库。

Git 添加操作添加文件到暂存区。

```
[jerry@CentOS project]$ git status -s
?? string
?? string.c

[jerry@CentOS project]$ git add string.c
```

Git是显示文件名前的问号。显然，这些文件不属于Git，Git 不知道该怎么用这些文件。这就是为什么Git是文件名前显示问号。

Jerry 添加文件到存储区域，git的状态命令将显示文件暂存区域。

```
[jerry@CentOS project]$ git status -s
A string.c
?? string
```

要提交更改他用git 的commit 命令-m选项。如果我们省略-m选项git会打开文本编辑器，在这里我们可以写多行提交信息。

```
[jerry@CentOS project]$ git commit -m 'Implemented my_strlen function'
```

上面的命令会产生以下结果。

```
[master cbe1249] Implemented my_strlen function
1 files changed, 24 insertions(+), 0 deletions(-)
create mode 100644 string.c
```

提交后查看日志信息，他使用 git 日志命令。它会显示提交ID所有提交的信息，提交作者，提交日期和提交的 SHA-1散列。

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit cbe1249b140dad24b2c35b15cc7e26a6f02d2277
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Implemented my_strlen function

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

Git 审查更改 - Git教程

但查看提交详细资料后，Jerry 实现字符串的长度不能为负数，所以他决定改变my_strlen函数的返回类型。

Jerry 使用git日志命令来查看日志信息。

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit cbe1249b140dad24b2c35b15cc7e26a6f02d2277
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Implemented my_strlen function
```

Jerry 使用git show命令查看提交的细节。 Git的show命令的SHA-1提交ID作为参数。

```
[jerry@CentOS project]$ git show cbe1249b140dad24b2c35b15cc7e26a6f02d2277
```

上面的命令会产生以下结果。

```
commit cbe1249b140dad24b2c35b15cc7e26a6f02d2277
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Implemented my_strlen function

diff --git a/string.c b/string.c
new file mode 100644
index 0000000..187afb9
--- /dev/null
+++ b/string.c
@@ -0,0 +1,24 @@
+#include <stdio.h>
+
+int my_strlen(char *s)
+{
+
char *p = s;
+
+
while (*p)
+ ++p;
+ return (p -s );
+}
+
```

他改变了函数的返回类型 从int 修改为 size_t。测试代码后，他查看其变化运行git diff命令。

```
[jerry@CentOS project]$ git diff
```

上面的命令会产生以下结果。

```
diff --git a/string.c b/string.c
index 187afb9..7da2992 100644
--- a/string.c
+++ b/string.c
@@ -1,6 +1,6 @@
#include <stdio.h>

-int my_strlen(char *s)
+size_t my_strlen(char *s)
{
char *p = s;
@@ -18,7 +18,7 @@ int main(void)
};
for (i = 0; i < 2; ++i)
- printf("string lenght of %s = %d
", s[i], my_strlen(s[i]));
+ printf("string lenght of %s = %lu
", s[i], my_strlen(s[i]));
return 0;
}
```

Git 的差异显示+号前行，这是新增加的，并显示符号被删除。

Git 提交更改 - Git教程

Jerry 已经提交的更改，他想纠正他的最后一次提交，在这种情况下，git 的修改将帮助操作。最后提交修改操作的变化，包括提交信息，它创建新的提交ID。

修改操作之前，他会检查提交日志。

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit cbe1249b140dad24b2c35b15cc7e26a6f02d2277
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Implemented my_strlen function

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

Jerry 提交了新的变化 - 修改操作，并查看提交日志。

```
[jerry@CentOS project]$ git status -s
M string.c
?? string

[jerry@CentOS project]$ git add string.c

[jerry@CentOS project]$ git status -s
M string.c
?? string

[jerry@CentOS project]$ git commit --amend -m 'Changed return type of my_strlen to size_t
[master d1e19d3] Changed return type of my_strlen to size_t
1 files changed, 24 insertions(+), 0 deletions(-)
create mode 100644 string.c
```

现在 git 的日志，将显示新的提交信息与新的提交ID

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

Git 推送操作 - Git教程

Jerry 修改了他的最后一次提交的修改操作，他已经准备好将更改。推操作的数据永久存储的Git 仓库。推操作成功后，其他开发人员可以看到Jerry 的变化。

他执行的git日志命令来查看提交的细节。

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t
```

push操作之前，他要审查他的变化，所以使用git show命令来查看他的变化。

```
[jerry@CentOS project]$ git show d1e19d316224cddc437e3ed34ec3c931ad803958
```

上面的命令会产生以下结果。

```
commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

diff --git a/string.c b/string.c
new file mode 100644
index 0000000..7da2992
--- /dev/null
+++ b/string.c
@@ -0,0 +1,24 @@
+#include <stdio.h>
+
+size_t my_strlen(char *s)
+{
+
char *p = s;
+
+
while (*p)
+ ++p;
+ return (p -s );
+}
+
+int main(void)
+{
+ int i;
+ char *s[] = {
+ "Git tutorials",
+ "Tutorials Yiibai"
+
};
+
+
+
for (i = 0; i < 2; ++i)
printf("string lenght of %s = %lu
", s[i], my_strlen(s[i]));
+
+
return 0;
+}
```

Jerry 为他的变化感到高兴，他是准备推他的变化。

```
[jerry@CentOS project]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 4, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 517 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
19ae206..d1e19d3 master −> master
```

Jerry 的变化成功地推到版本库，现在其他开发人员可以查看他的变化进行克隆或更新操作。

Git 更新操作 - Git教程

修改现有函数

Tom 执行克隆操作后，看到新的文件string.c，他想知道这个文件到存储库？目的是什么？于是，他执行 git 日志命令。

```
[tom@CentOS ~]$ git clone gituser@git.server.com:project.git
```

上面的命令会产生以下结果。

```
Initialized empty Git repository in /home/tom/project/.git/
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 100% (6/6), 726 bytes, done.
remote: Total 6 (delta 0), reused 0 (delta 0)
```

克隆操作将当前的工作目录内创建新的目录。他改变目录到新创建的目录和执行 git日志命令。

```
[tom@CentOS ~]$ cd project/

[tom@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

查看记录后，他意识到，string.c 文件加入Jerry 实现基本字符串操作。他是好奇Jerry 的代码。于是他打开文本编辑string.c 文件，并立即找到了一个漏洞。 my_strlen函数中 Jerry 没有使用常量指针。于是，他决定修改Jerry 的代码。修改后的代码会这个样子。

```
[tom@CentOS project]$ git diff
```

上面的命令会产生以下结果。

```
diff --git a/string.c b/string.c
index 7da2992..32489eb 100644
--- a/string.c
+++ b/string.c
@@ -1,8 +1,8 @@
#include <stdio.h>-size_t my_strlen(char *s)
+size_t my_strlen(const char *s)
{
- char *p = s;
+ const char *p = s;
while (*p)
++p;</stdio.h>
```

经过测试，他提交他的变化。

```
[tom@CentOS project]$ git status -s
M string.c
?? string

[tom@CentOS project]$ git add string.c

[tom@CentOS project]$ git commit -m 'Changed char yiibaier to const char yiibaier'
[master cea2c00] Changed char yiibaier to const char yiibaier
1 files changed, 2 insertions(+), 2 deletions(-)

[tom@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit cea2c000f53ba99508c5959e3e12fff493b
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 08:32:07 2013 +0530

Changed char yiibaier to const char yiibaier

commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530
Initial commit
```

Tom 使用git push命令，推他的变化。

```
[tom@CentOS project]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 336 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
d1e19d3..cea2c00 master −> master
```

添加新函数

与此同时，Jerry 决定实现字符串比较函数。于是，他修改string.c文件。修改后的文件会这个样子。

```
[jerry@CentOS project]$ git diff
```

上面的命令会产生以下结果。

```
index 7da2992..bc864ed 100644
--- a/string.c
+++ b/string.c
30Git Tutorials
@@ -9,9 +9,20 @@ size_t my_strlen(char *s)
return (p -s );
}
+char *my_strcpy(char *t, char *s)
+{
+
char *p = t;
+
+ while (*t++ = *s++)
+ ;
+
+
return p;
+}
+
int main(void)
{
int i;
+
char p1[32];
char *s[] = {
"Git tutorials",
"Tutorials Yiibai"
@@ -20,5 +31,7 @@ int main(void)
for (i = 0; i < 2; ++i)
printf("string lenght of %s = %lu
", s[i], my_strlen(s[i]));
+
printf("%s
", my_strcpy(p1, "Hello, World !!!"));
+
return 0;
}
```

经过测试，他准备推他的变化。

```
[jerry@CentOS project]$ git status -s
M string.c
?? string

[jerry@CentOS project]$ git add string.c

[jerry@CentOS project]$ git commit -m "Added my_strcpy function"
[master e944e5a] Added my_strcpy function
1 files changed, 13 insertions(+), 0 deletions(-)
```

push操作之前，他验证通过查看日志信息提交。

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit e944e5aab74b26e7447d3281b225309e4e59efcd
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:41:42 2013 +0530

Added my_strcpy function

commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530

Initial commit
```

Jerry 为变化感到高兴，他想推他的变化

```
[jerry@CentOS project]$ git push origin master
```

上面的命令会产生以下结果。

```
To gituser@git.server.com:project.git
! [rejected]
master −> master (non-fast-forward)
error: failed to push some refs to 'gituser@git.server.com:project.git'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes before pushing again. See the 'Note about
fast-forwards' section of 'git push --help' for details.
```

但是，Git 是不允许Jerry 推他的变化。因为Git确定该远程仓库和Jerry 的本地资源库不同步。正因为如此，他可能会失去项目的历史。因此，为了避免这种混乱的Git 对此操作失败。Jerry 必须首先更新它的本地存储库，然后再只有他才能把他自己的变化。

取最新变化

Jerry 执行git pull命令远程命令来同步自己的本地仓库。

```
[jerry@CentOS project]$ git pull
```

上面的命令会产生以下结果。

```
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git.server.com:project
d1e19d3..cea2c00 master -> origin/master
First, rewinding head to replay your work on top of it...
Applying: Added my_strcpy function
```

抽取操作后Jerry 检查日志消息，并找到 Tom 的提交详细，提交ID为 cea2c000f53ba99508c5959e3e12fff493ba6f69

```
[jerry@CentOS project]$ git log
```

上面的命令会产生以下结果。

```
commit e86f0621c2a3f68190bba633a9fe6c57c94f8e4f
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:41:42 2013 +0530

Added my_strcpy function

commit cea2c000f53ba99508c5959e3e12fff493ba6f69
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 08:32:07 2013 +0530

Changed char yiibaier to const char yiibaier

commit d1e19d316224cddc437e3ed34ec3c931ad803958
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 08:05:26 2013 +0530

Changed return type of my_strlen to size_t

commit 19ae20683fc460db7d127cf201a1429523b0e319
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 07:32:56 2013 +0530
Initial commit
```

现在，Jerry 的本地存储库是完全同步的远程仓库。所以，他可以放心地将他的变化。

```
[jerry@CentOS project]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 455 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
cea2c00..e86f062 master -> master
```

Git 藏匿操作 - Git教程

假设您正在为您的产品实施的一项新功能。你的代码是在推进开发进度而客户不断升级需求突然来了。正因为如此，你必须保持放下你的新功能，工作几个小时。你不能提交你的部分代码，也不能扔掉你的变化。所以，你需要一些临时空间，在那里你可以存储你的部分修改，以便以后再提交。

在Git中，藏匿操作需要修改的跟踪文件和阶段的变化，并将其保存在栈上未完成的更改，可以在任何时候重新。

```
[jerry@CentOS project]$ git status -s
M string.c
?? string
```

现在要切换分支机构为客户不断升级，但你不想提交你的工作，所以你会藏匿的变化。要推一个新的藏匿到您的堆栈，运行git stash命令

```
[jerry@CentOS project]$ git stash
Saved working directory and index state WIP on master: e86f062 Added my_strcpy function
HEAD is now at e86f062 Added my_strcpy function
```

现在你的工作目录是干净的，所有的改变都保存在堆栈。让我们用git status命令验证。

```
[jerry@CentOS project]$ git status -s
?? string
```

现在可以安全地切换分支和做其他工作。我们可以看到的藏匿的变化列表通过使用 git stash list 命令。

```
[jerry@CentOS project]$ git stash list
stash@{0}: WIP on master: e86f062 Added my_strcpy function
```

假设你解决了客户不断升级和你要回到你的工作，已经做了一半的代码。只要执行git stash pop 命令，它会从堆栈中删除的变化，并把它放在当前工作目录。

```
[jerry@CentOS project]$ git status -s
?? string

[jerry@CentOS project]$ git stash pop
```

上面的命令会产生以下结果。

```
# On branch master
# Changed but not updated:
# (use "git add <file>..." to update what will be committed)
# (use "git checkout -- <file>..." to discard changes in working directory)
#
#
modified: string.c
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#
string
no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (36f79dfedae4ac20e2e8558830154bd6315e72d4)

[jerry@CentOS project]$ git status -s
M string.c
?? string</file></file></file>
```

Git 移动操作 - Git教程

顾名思义移动(move)操作移动目录或文件从一个位置到另一个。Tom 决定移动到src目录下的源代码。因此，修改后的目录结构看起来会像这样。

```
[tom@CentOS project]$ pwd
/home/tom/project

[tom@CentOS project]$ ls
README string string.c

[tom@CentOS project]$ mkdir src

[tom@CentOS project]$ git mv string.c src/

[tom@CentOS project]$ git status -s
R string.c -> src/string.c
?? string
```

要进行这些永久性更改，以便其他开发人员可以看到这一点，我们必须修改的目录结构推到远程存储库。

```
[tom@CentOS project]$ git commit -m "Modified directory structure"

[master 7d9ea97] Modified directory structure
1 files changed, 0 insertions(+), 0 deletions(-)
rename string.c => src/string.c (100%)

[tom@CentOS project]$ git push origin master
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
e86f062..7d9ea97 master -> master
```

在Jerry 的本地资源库，抽取操作前，它会显示旧的目录结构。

```
[jerry@CentOS project]$ pwd
/home/jerry/jerry_repo/project

[jerry@CentOS project]$ ls
README string string.c
```

但是，抽取(pull)操作后的目录结构将得到更新。现在，Jerry 可以看到该目录内的 src目录和文件。

```
[jerry@CentOS project]$ git pull
remote: Counting objects: 4, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From git.server.com:project
e86f062..7d9ea97 master −> origin/master
First, rewinding head to replay your work on top of it...
Fast-forwarded master to 7d9ea97683da90bcdb87c28ec9b4f64160673c8a.

[jerry@CentOS project]$ ls
README src string

[jerry@CentOS project]$ ls src/
string.c
```

Git 重命名操作 - Git教程

截至目前，Tome 和Jerry 都使用手动命令来编译自己的项目。Jerry 决定为他们的项目创建 Makefile，并给予适当的名称来命名“string.c” 文件。

```
[jerry@CentOS project]$ pwd
/home/jerry/jerry_repo/project

[jerry@CentOS project]$ ls
README src

[jerry@CentOS project]$ cd src/

[jerry@CentOS src]$ git add Makefile

[jerry@CentOS src]$ git mv string.c string_operations.c

[jerry@CentOS src]$ git status -s
A Makefile
R string.c -> string_operations.c
```

Git 是显示R在文件之前名称来指示文件已更名。

对于提交操作Jerry 使用 -a标志，这使得git 提交自动检测修改过的文件。

```
[jerry@CentOS src]$ git commit -a -m 'Added Makefile and renamed strings.c to
string_operations.c '

[master 94f7b26] Added Makefile and renamed strings.c to string_operations.c
1 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/Makefile
rename src/{string.c => string_operations.c} (100%)
```

提交后，他推送了他的修改到版本库。

```
[jerry@CentOS src]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 6, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 396 bytes, done.
Total 4 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
7d9ea97..94f7b26 master -> master
```

现在，其他开发人员可以通过更新他们的本地资源库中的这些修改。

Git 删除操作 - Git教程

Tom 更新了自己的本地存储库并进入src目录下找到编译后的二进制。查看提交信息后，他意识到，编译后的二进制是由Jerry加入的。 .

```
[tom@CentOS src]$ pwd
/home/tom/project/src

[tom@CentOS src]$ ls
Makefile string_operations string_operations.c

[tom@CentOS src]$ file string_operations
string_operations: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linke
shared libs), for GNU/Linux 2.6.18, not stripped

[tom@CentOS src]$ git log
commit 29af9d45947dc044e33d69b9141d8d2dad37cc62
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 10:16:25 2013 +0530

Added compiled binary
```

VCS用于存储源代码，而不是只对可执行的二进制文件。因此，Tom 决定从资源库中删除此文件。对于进一步的操作，他使用git 的 rm命令。

```
[tom@CentOS src]$ ls
Makefile string_operations string_operations.c

[tom@CentOS src]$ git rm string_operations
rm 'src/string_operations'

[tom@CentOS src]$ git commit -a -m "Removed executable binary"

[master 5776472] Removed executable binary
1 files changed, 0 insertions(+), 0 deletions(-)
delete mode 100755 src/string_operations
```

提交后，他推送了他的修改到版本库。

```
[tom@CentOS src]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 310 bytes, done.
Total 3 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
29af9d4..5776472 master −> master
```

Git 修正错误 - Git教程

大部分的人都会犯错。所以每VCS提供了一个功能，修正错误，直到特定的点。 Git提供功能使用，我们可以撤销已作出的修改到本地资源库。

假设用户不小心做了一些更改，以他的本地的仓库，现在他要扔掉这些变化。在这种情况下，恢复操作中起着重要的作用。

恢复未提交的更改

让我们假设Jerry 不小心修改文件从自己的本地仓库。但他想扔掉他的修改。要处理这种情况，我们可以使用git checkout命令。我们可以使用这个命令来恢复文件的内容。

```
[jerry@CentOS src]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git status -s
M string_operations.c

[jerry@CentOS src]$ git checkout string_operations.c

[jerry@CentOS src]$ git status –s
```

甚至我们可以使用git checkout命令删除的文件从本地库。让我们假设Tom 删除文件从本地存储库，我们希望这个文件。我们可以做到这一点，使用相同的命令。

```
[tom@CentOS src]$ pwd
/home/tom/top_repo/project/src

[tom@CentOS src]$ ls -1
Makefile
string_operations.c

[tom@CentOS src]$ rm string_operations.c

[tom@CentOS src]$ ls -1
Makefile

[tom@CentOS src]$ git status -s
D string_operations.c
```

Git是显示文件名前的字母D。这标志着该文件已被删除，从本地资源库。

```
[tom@CentOS src]$ git checkout string_operations.c

[tom@CentOS src]$ ls -1
Makefile
string_operations.c

[tom@CentOS src]$ git status -s
```

> 注意：我们可以执行所有这些操作之前提交操作。

删除临时区域变化

我们已经看到，当我们执行加法运算;文件移动从本地存储库，参数区域。如果用户不小心修改一个文件，并把它添加到临时区域，但他马上意识到犯了错误。他希望恢复他的改变。我们可以处理这种情况下，通过使用git checkout命令。

在Git是一个HEAD指针始终指向最新提交。如果想撤销变更分阶段区域，那么可以使用git checkout命令，但checkout命令，必须提供额外的参数HEAD指针。额外提交指针参数指示的git checkout命令，重置工作树，还能够去除分阶段。

让我们假设Tom 从自己的本地仓库修改一个文件。如果我们查看这个文件的状态，它会显示文件被修改，但不加入临时区域。

```
tom@CentOS src]$ pwd
/home/tom/top_repo/project/src
# Unmodified file

[tom@CentOS src]$ git status -s

# Modify file and view it’s status.
[tom@CentOS src]$ git status -s
M string_operations.c

[tom@CentOS src]$ git add string_operations.c
```

Git 的状态显示该文件是在分期区域，现在它恢复使用git checkout命令和视图状态恢复的文件。

```
[tom@CentOS src]$ git checkout HEAD -- string_operations.c

[tom@CentOS src]$ git status -s
```

将HEAD指针与GIT复位

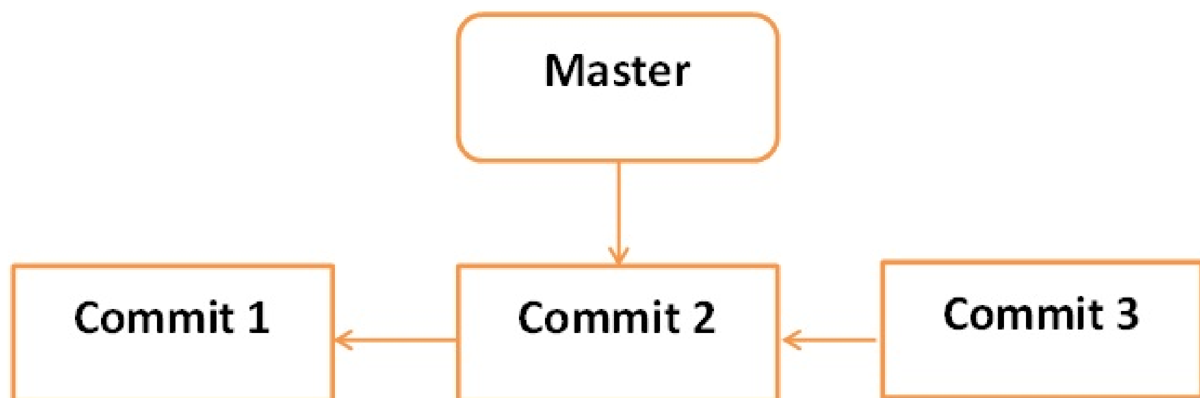
做一些改变后，可能会决定删除这些更改。 Git 复位命令是用来重置或恢复一些变化。我们可以执行三种不同类型的复位操作。

下图显示图形表示 Git复位命令。



Before git reset command

Yiibai.com



After git reset command

Yiibai.com

SOFT

每个分支都有HEAD 指针指向最新提交。如果我们使用git --soft复位命令选项，随后提交ID，然后将只有头指针复位，不破坏任何东西。

.git/refs/heads/master 文件存储的提交ID 的HEAD 指针。我们可以验证它通过使用git log -1 命令。

```
[jerry@CentOS project]$ cat .git/refs/heads/master
577647211ed44fe2ae479427a0668a4f12ed71a1
```

现在查看最新提交的ID，这将配合上述提交ID。

```
[jerry@CentOS project]$ git log -2
```

上面的命令会产生以下结果。

```
commit 577647211ed44fe2ae479427a0668a4f12ed71a1
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 10:21:20 2013 +0530

Removed executable binary

commit 29af9d45947dc044e33d69b9141d8d2dad37cc62
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 10:16:25 2013 +0530

Added compiled binary
```

让我们重设HEAD 指针。

```
[jerry@CentOS project]$ git reset --soft HEAD~
```

现在我们只重设HEAD 指针回到一个位置。让我们检查内容.git/refs/heads/master 文件.

```
[jerry@CentOS project]$ cat .git/refs/heads/master
29af9d45947dc044e33d69b9141d8d2dad37cc62
```

从文件提交ID改变，现在验证通过查看提交的信息。

```
jerry@CentOS project]$ git log -2
```

上面的命令会产生以下结果。

```
commit 29af9d45947dc044e33d69b9141d8d2dad37cc62
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 10:16:25 2013 +0530

Added compiled binary

commit 94f7b26005f856f1a1b733ad438e97a0cd509c1a
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 10:08:01 2013 +0530

Added Makefile and renamed strings.c to string_operations.c
```

混合

Git的复位 -- mixed选项从分段区域尚未提交还原更改。它仅恢复变化形成暂存区。实际所做的更改到工作副本的文件不受影响。默认的Git的复位相当于git的复位 -- mixed。

有关更多详细信息，请参阅部分删除同一章临时区域变化。

HARD

如果使用 - hard选项用 Git复位命令，它会清除暂存区域，它会重设HEAD 指针，以最后一次提交的具体提交ID，也删除本地文件的变化。

让我们检查提交的ID

```
[jerry@CentOS src]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git log -1
```

上面的命令会产生以下结果。

```
commit 577647211ed44fe2ae479427a0668a4f12ed71a1
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 10:21:20 2013 +0530

Removed executable binary
```

Jerry 修改过的文件，在文件的开始，加入单行注释。

```
[jerry@CentOS src]$ head -2 string_operations.c
/* This line be removed by git reset operation */
#include <stdio.h>
```

他使用git status命令验证。

```
[jerry@CentOS src]$ git status -s
M string_operations.c
```

Jerry 修改后的文件添加到分期区域，并验证它与git的状态运行。

```
[jerry@CentOS src]$ git add string_operations.c
[jerry@CentOS src]$ git status
```

上面的命令会产生以下结果。

```
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
#
modified: string_operations.c
#
```

Git的状态，显示该文件是在临时区域。现在重置HEAD 用 --hard选项。

```
[jerry@CentOS src]$ git reset --hard 577647211ed44fe2ae479427a0668a4f12ed71a1

HEAD is now at 5776472 Removed executable binary
```

Git 复位命令成功，这将恢复从分段区的文件，以及删除本地对文件所做的更改。

```
[jerry@CentOS src]$ git status -s
```

Git 状态显示，文件恢复从分段区。

```
[jerry@CentOS src]$ head -2 string_operations.c
#include <stdio.h>
```

Head 命令还显示，复位操作删除局部变化。

Git 标签操作 - Git教程

允许有意义的名称到一个特定的版本库中的标签操作。Tom 决定标记他们的项目代码，以便他们以后可以更容易访问。

创建标签

让我们标记当前HEAD使用git tag命令。他提供的标记名称前加上-a选项，使用-m选项，并提供标签信息。

```
tom@CentOS project]$ pwd
/home/tom/top_repo/project

[tom@CentOS project]$ git tag -a 'Release_1_0' -m 'Tagged basic string operation code' HE
```

如果想标记特定提交然后使用适当的COMMIT ID，而不是HEAD 指针。 Tom使用下面的命令推到远程存储库中的标签。

```
[tom@CentOS project]$ git push origin tag Release_1_0
```

上面的命令会产生以下结果。

```
Counting objects: 1, done.
Writing objects: 100% (1/1), 183 bytes, done.
Total 1 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
* [new tag]
Release_1_0 -> Release_1_0
```

查看标签

Tom 创建标签。现在，Jerry 可以查看所有可用标签通过使用Git tag命令使用-l选项。

```
[jerry@CentOS src]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git pull
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (1/1), done.
From git.server.com:project
* [new tag]
Release_1_0 -> Release_1_0
Current branch master is up to date.

[jerry@CentOS src]$ git tag -l
Release_1_0
```

Jerry 使用Git的show命令后跟标记名称的有关标签查看更多细节。

```
[jerry@CentOS src]$ git show Release_1_0
```

上面的命令会产生以下结果。

```
tag Release_1_0
Tagger: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 13:45:54 2013 +0530

Tagged basic string operation code

commit 577647211ed44fe2ae479427a0668a4f12ed71a1
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 10:21:20 2013 +0530

Removed executable binary

diff --git a/src/string_operations b/src/string_operations
deleted file mode 100755
index 654004b..0000000
Binary files a/src/string_operations and /dev/null differ
```

删除标签

Tom使用下面的命令来删除标记从本地以及远程仓库。

```
[tom@CentOS project]$ git tag
Release_1_0

[tom@CentOS project]$ git tag -d Release_1_0
Deleted tag 'Release_1_0' (was 0f81ff4)
# Remove tag from remote repository.

[tom@CentOS project]$ git push origin :Release_1_0
To gituser@git.server.com:project.git
- [deleted]
Release_1_0
```

Git 补丁操作 - Git教程

补丁是文本文件，其内容是相似于Git diff，但随着代码，它也有元数据有关提交，如提交ID，日期，提交信息等，我们可以创建补丁提交和其他人可以将它们应用到自己的资料库。

Jerry 为他们的项目实现strcat函数。 Jerry 可以创建自己的代码路径发送到Tom。那么他就可以收到Jerry 的代码补丁。

杰里使用Git format-patch 命令来创建最新提交的补丁。如果想创建补丁具体提交，然后使用 COMMIT_ID 和 ormat-patch 命令。

```
[jerry@CentOS project]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git status -s
M string_operations.c
?? string_operations

[jerry@CentOS src]$ git add string_operations.c

[jerry@CentOS src]$ git commit -m "Added my_strcat function"

[master b4c7f09] Added my_strcat function
1 files changed, 13 insertions(+), 0 deletions(-)

[jerry@CentOS src]$ git format-patch -1
0001-Added-my_strcat-function.patch
```

上面的命令创建 .patch文件里在当前工作目录。 Tom可以使用这个补丁修改他的文件。 Git提供两个命令来应用补丁调幅分别为： git am 和 git apply . Git apply命令修改本地文件时，而无需创建提交，git am命令修改文件，会一并创建提交。

适用于修补程序并创建提交使用下面的命令。

```
[tom@CentOS src]$ pwd
/home/tom/top_repo/project/src

[tom@CentOS src]$ git diff

[tom@CentOS src]$ git status –s

[tom@CentOS src]$ git apply 0001-Added-my_strcat-function.patch

[tom@CentOS src]$ git status -s
M string_operations.c
?? 0001-Added-my_strcat-function.patch
```

补丁得到成功应用，现在我们可以使用git diff命令查看修改。

```
[tom@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
diff --git a/src/string_operations.c b/src/string_operations.c
index 8ab7f42..f282fcf 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,5 +1,16 @@
#include <stdio.h>
+char *my_strcat(char *t, char *s)
diff --git a/src/string_operations.c b/src/string_operations.c
index 8ab7f42..f282fcf 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,5 +1,16 @@
#include <stdio.h>
+char *my_strcat(char *t, char *s)
+{
+
char *p = t;
+
+
+
while (*p)
++p;
+
while (*p++ = *s++)
+ ;
+ return t;
+}
+
size_t my_strlen(const char *s)
{
const char *p = s;
@@ -23,6 +34,7 @@ int main(void)
{
```

Git 管理分支 - Git教程

分支操作可以创造另一条线的发展。对fork过程分为两个不同的方向发展，我们可以使用此操作。例如，我们发布了6.0版本的产品，我们可能要创建一个分支，使7.0功能的发展可以保持独立从6.0 bug修复。

创建分支

使用Git分支<branch name> 命令创建新的分支。从现有的，我们可以创建一个新的分支。我们可以使用特定的提交或标签作为一个起点。如果没有提供任何具体的提交ID，然后分支将HEAD 创建作为一个起点。

```
[jerry@CentOS src]$ git branch new_branch

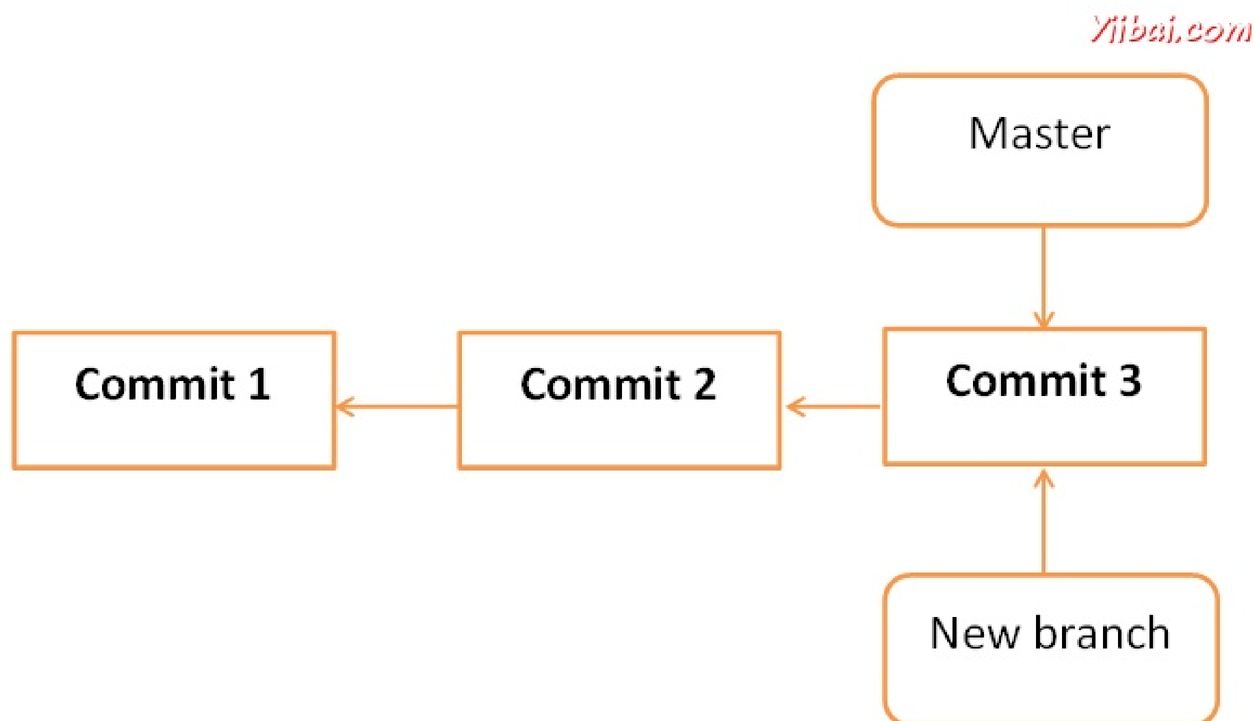
[jerry@CentOS src]$ git branch
* master
new_branch
```

创建新的分支，Tom用 git branch命令列出可用的分支。 Git会显示星号标记之前，当前检出的分支。

下面是创建分支操作的图形表示



Before create branch command



After create branch operation

Yiibai.com

切换分支

Jerry 使用git checkout命令到分支之间切换。

```
[jerry@CentOS src]$ git checkout new_branch
Switched to branch 'new_branch'
[jerry@CentOS src]$ git branch
master
* new_branch
```

创建和切换分支的快捷方式

在上面的例子中，我们使用了两个命令来创建和切换分支。 Git提供checkout命令 -b选项，此操作将创建新的分支，并立即切换到新的分支。

```
[jerry@CentOS src]$ git checkout -b test_branch
Switched to a new branch 'test_branch'

[jerry@CentOS src]$ git branch
master
new_branch
* test_branch
```

删除分支

一个分支可以用git branch命令的-D选项被删除。但在此之前， 删除现有的分支切换到其他分支。

Jerry 当前在test_branch 想要删除该分支。于是，他分支和删除分支切换，如下图所示。

```
[jerry@CentOS src]$ git branch
master
new_branch
* test_branch

[jerry@CentOS src]$ git checkout master
Switched to branch 'master'

[jerry@CentOS src]$ git branch -D test_branch
Deleted branch test_branch (was 5776472).
```

现在Git会显示只有两个分支。

```
[jerry@CentOS src]$ git branch
* master
new_branch
```

重命名分支

Jerry 决定添加宽字符支持他的字符串操作项目。他已经创建了一个新的分支，但分支名称是不恰当的。于是，他通过使用-m选项，其次是旧分支名称和新分支名称变更分支名称。

```
[jerry@CentOS src]$ git branch
* master
new_branch

[jerry@CentOS src]$ git branch -m new_branch wchar_support
```

现在git branch命令将显示新分支名称。

```
[jerry@CentOS src]$ git branch
* master
wchar_support
```

合并两个分支

Jerry 实现函数返回字符串的长度为宽字符串。新代码将看起来像这样

```
[jerry@CentOS src]$ git branch
master
* wchar_support

[jerry@CentOS src]$ pwd
/home/jerry/jerry_repo/project/src

[jerry@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
t a/src/string_operations.c b/src/string_operations.c
index 8ab7f42..8fb4b00 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,4 +1,14 @@
#include <stdio.h>
+#include <wchar.h>
+
+size_t w_strlen(const wchar_t *s)
+{
+
const wchar_t *p = s;
+
+
while (*p)
+ ++p;
+ return (p - s);
+}
```

测试后，他提交他的变化，并推到新的分支。

```
[jerry@CentOS src]$ git status -s
M string_operations.c
?? string_operations

[jerry@CentOS src]$ git add string_operations.c

[jerry@CentOS src]$ git commit -m 'Added w_strlen function to return string lenght of wch
string'

[wchar_support 64192f9] Added w_strlen function to return string lenght of wchar_t string
1 files changed, 10 insertions(+), 0 deletions(-)
```

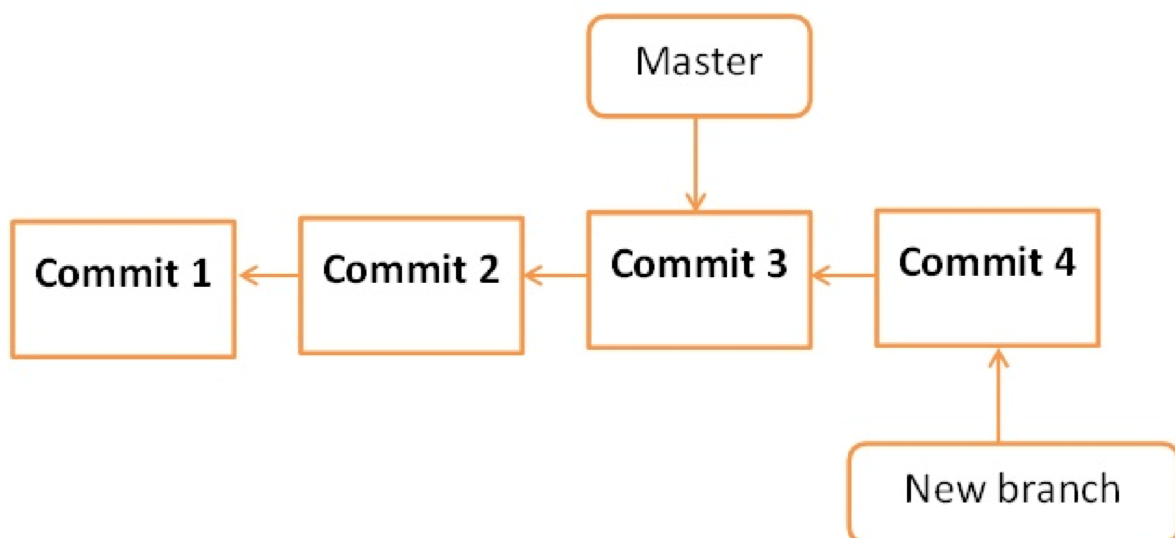
注意杰里推动这些变化的新分支，这就是为什么他用wchar_support分支的名称，而不是master分支。

```
[jerry@CentOS src]$ git push origin wchar_support **<------------- Observer branch_name**
```

上面的命令会产生以下结果。

```
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 507 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
* [new branch]
wchar_support -> wchar_support
```

经过分支提交的变化，新分支会这个样子。



After commit in new branch

Tom 好奇Jerry 在做什么在他的私人分支，这就是为什么他检查日志从wchar_support 分支。

```
[tom@CentOS src]$ pwd
/home/tom/top_repo/project/src

[tom@CentOS src]$ git log origin/wchar_support -2
```

上面的命令会产生以下结果。

```
commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3
Author: Jerry Mouse <jerry@yiibai.com>
Date: Wed Sep 11 16:10:06 2013 +0530

Added w_strlen function to return string lenght of wchar_t string

commit 577647211ed44fe2ae479427a0668a4f12ed71a1
Author: Tom Cat <tom@yiibai.com>
Date: Wed Sep 11 10:21:20 2013 +0530

Removed executable binary
```

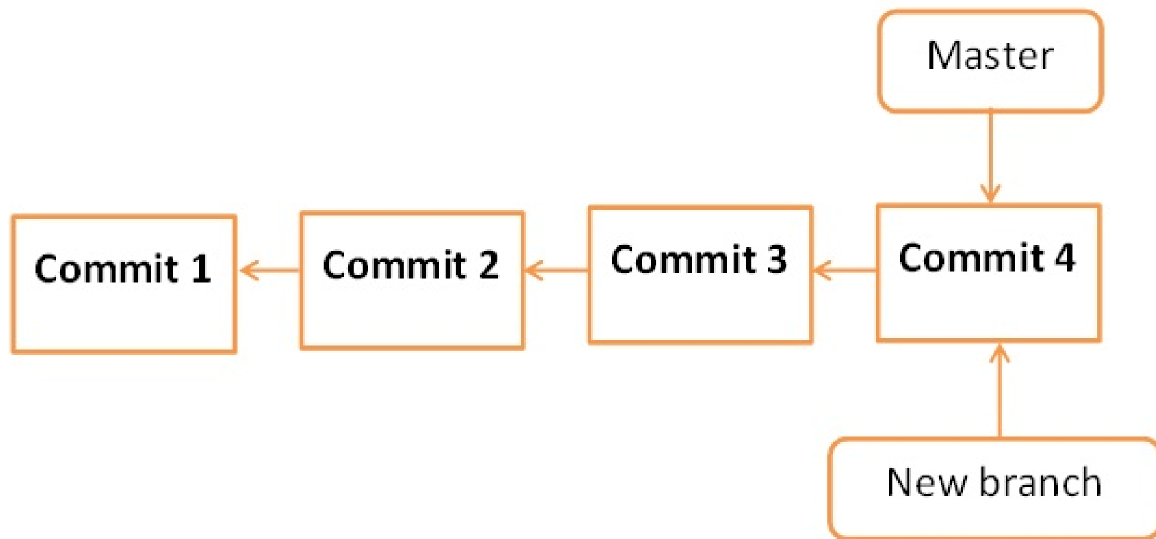
通过查看提交的信息，Tom 意识到Jerry 实现宽字符strlen 函数，他希望同样的功能集成到主分支。而不是重新实现他的分支合并到主分支，他决定采用杰里的代码。

```
[tom@CentOS project]$ git branch
* master

[tom@CentOS project]$ pwd
/home/tom/top_repo/project

[tom@CentOS project]$ git merge origin/wchar_support
Updating 5776472..64192f9
Fast-forward
src/string_operations.c | 10 ++++++++++
1 files changed, 10 insertions(+), 0 deletions(-)
```

合并操作后的主分支会这个样子。



现在wchar_support分支合并到主分支中我们可以验证它的查看提交信息，通过查看修改成string_operation.c文件。

```
[tom@CentOS project]$ cd src/

[tom@CentOS src]$ git log -1

commit 64192f91d7cc2bcdf3bf946dd33ece63b74184a3
Author: Jerry Mouse <jerry@yiibai.com>Date: Wed Sep 11 16:10:06 2013 +0530

Added w_strlen function to return string lenght of wchar_t string

[tom@CentOS src]$ head -12 string_operations.c</jerry@yiibai.com>
```

上面的命令会产生以下结果。

```
#include <stdio.h>
#include <wchar.h>
size_t w_strlen(const wchar_t *s)
{
   const wchar_t *p = s;

   while (*p)
      ++p;

   return (p - s);
}
```

经过测试，他把他的代码更改到主分支。

```
[tom@CentOS src]$ git push origin master
Total 0 (delta 0), reused 0 (delta 0)
To gituser@git.server.com:project.git
5776472..64192f9 master −> master
```

重订分支

Git 的 rebase命令的一个分支合并的命令，但不同的是，它修改提交的顺序。

Git merge命令，试图把从其他分支提交当前的本地分支的HEAD上。例如 本地的分支已经提交A-> B-> C-> D和合并分支已提交A-> B-> X> Y，则Git合并将当前转换像这样的本地分行A-> B-> C-> D-> X-> Y

Git 的rebase命令试图找到当前的本地分支和合并分支之间的共同祖先。然后把修改提交的顺序，在当前的本地分支提交中的本地分支。例如，如果当地的分支已提交A-> B-> C-> D和合并分支已提交A-> B-> X-> Y，Git衍合的类似A-> B转换成当前的本地分支A−>B−>X−>Y−>C−>D

当多个开发人员在一个单一的远程资源库的工作，你不能在远程仓库提交修改订单。在这种情况下，可以使用变基操作把本地提交的远程仓库之上的提交，可以推送这些变化。

Git 冲突处理 - Git教程

执行wchar_support分支变化

Jerry 工作在wchar_support分支。他改变了名称的功能和测试后，他提交他的变化。

```
[jerry@CentOS src]$ git branch
 master
* wchar_support
[jerry@CentOS src]$ git diff
```

上面的命令产生以下结果

```
diff --git a/src/string_operations.c b/src/string_operations.c
index 8fb4b00..01ff4e0 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,7 +1,7 @@
#include <stdio.h>
#include <wchar.h>
-size_t w_strlen(const wchar_t *s)
+size_t my_wstrlen(const wchar_t *s)
{
const wchar_t *p = s;
```

验证代码后，他提交了他的变化。

```
[jerry@CentOS src]$ git status -s
M string_operations.c

[jerry@CentOS src]$ git add string_operations.c

[jerry@CentOS src]$ git commit -m 'Changed function name'
[wchar_support 3789fe8] Changed function name
1 files changed, 1 insertions(+), 1 deletions(-)

[jerry@CentOS src]$ git push origin wchar_support
```

上面的命令会产生以下结果。

```
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 409 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
64192f9..3789fe8 wchar_support -> wchar_support
```

在主分支进行更改

同时在主分支Tom 也改变了名称相同的功能，并将其更改到主分支。

```
[tom@CentOS src]$ git branch
* master
[tom@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
diff --git a/src/string_operations.c b/src/string_operations.c
index 8fb4b00..52bec84 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,7 +1,8 @@
#include <stdio.h>
#include <wchar.h>
-size_t w_strlen(const wchar_t *s)
+/* wide character strlen fucntion */
+size_t my_wc_strlen(const wchar_t *s)
{
const wchar_t *p = s;
```

验证差异后，他提交了他的变化。

```
[tom@CentOS src]$ git status -s
M string_operations.c

[tom@CentOS src]$ git add string_operations.c

[tom@CentOS src]$ git commit -m 'Changed function name from w_strlen to my_wc_strlen'
[master ad4b530] Changed function name from w_strlen to my_wc_strlen
1 files changed, 2 insertions(+), 1 deletions(-)

[tom@CentOS src]$ git push origin master
```

上面的命令会产生以下结果。

```
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 470 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
64192f9..ad4b530 master -> master
```

strchr 函数在分支Jerry 实现 wchar_support宽字符字符串。经过测试，他提交和推送其变化 wchar_support 分支。

```
[jerry@CentOS src]$ git branch
master
* wchar_support
[jerry@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
diff --git a/src/string_operations.c b/src/string_operations.c
index 01ff4e0..163a779 100644
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@ -1,6 +1,16 @@
#include <stdio.h>
#include <wchar.h>
+wchar_t *my_wstrchr(wchar_t *ws, wchar_t wc)
+{
+
while (*ws) {
+
if (*ws == wc)
+
return ws;
+
++ws;
+ }
+ return NULL;
+}
+
size_t my_wstrlen(const wchar_t *s)
{
const wchar_t *p = s;
```

验证变化后，他提交他的变化。

```
[jerry@CentOS src]$ git status -s
M string_operations.c

[jerry@CentOS src]$ git add string_operations.c

[jerry@CentOS src]$ git commit -m 'Addded strchr function for wide character string'
[wchar_support 9d201a9] Addded strchr function for wide character string
1 files changed, 10 insertions(+), 0 deletions(-)

[jerry@CentOS src]$ git push origin wchar_support
```

上面的命令会产生以下结果。

```
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 516 bytes, done.
Total 4 (delta 1), reused 0 (delta 0)
To gituser@git.server.com:project.git
3789fe8..9d201a9 wchar_support -> wchar_support
```

对付冲突

Tom想看，Jerry 在他的私人分支做什么？这就是为什么他试图从wchar_support分支把最新的修改，但Git 放弃操作在得到错误消息后。

```
[tom@CentOS src]$ git pull origin wchar_support
```

上面的命令会产生以下结果。

```
remote: Counting objects: 11, done.
63Git Tutorials
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 2), reused 0 (delta 0)
Unpacking objects: 100% (8/8), done.
From git.server.com:project
* branch
wchar_support -> FETCH_HEAD
Auto-merging src/string_operations.c
**CONFLICT (content): Merge conflict in src/string_operations.c**
Automatic merge failed; fix conflicts and then commit the result.
```

解决冲突

从错误消息很显然知道，是有冲突的在src/string_operations.c。他运行 git diff 命令查看进一步的细节。

```
[tom@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
diff --cc src/string_operations.c
index 52bec84,163a779..0000000
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@@ -1,8 -1,17 +1,22 @@@
#include <stdio.h>
#include <wchar.h>
++<<<<<<< HEAD
+/* wide character strlen fucntion */
+size_t my_wc_strlen(const wchar_t *s)
++=======
+ wchar_t *my_wstrchr(wchar_t *ws, wchar_t wc)
+{
+
+
while (*ws) {
if (*ws == wc)
+
return ws;
+
++ws;
+ }
+ return NULL;
+}
+
+ size_t my_wstrlen(const wchar_t *s)
++>>>>>>>9d201a9c61bc4713f4095175f8954b642dae8f86
{
const wchar_t *p = s;
```

由于Tom 和Jerry 在相同的功能更改的名称，Git不知道如何去做，因此这就是为什么它要求用户手动解决冲突。

Tom 决定保用 Jerry 建议的函数名，但他使用原来注释，因为这是他加入的。删除冲突标记混帐后差异会看起来像这样。

```
[tom@CentOS src]$ git diff
```

上面的命令会产生以下结果。

```
diff --cc src/string_operations.c
diff --cc src/string_operations.c
index 52bec84,163a779..0000000
--- a/src/string_operations.c
+++ b/src/string_operations.c
@@@ -1,8 -1,17 +1,18 @@@
#include <stdio.h>
#include <wchar.h>
+ wchar_t *my_wstrchr(wchar_t *ws, wchar_t wc)
+{
+
while (*ws) {
+
if (*ws == wc)
+
return ws;
+
++ws;
+ }
+ return NULL;
+}
+
+/* wide character strlen fucntion */
- size_t my_wc_strlen(const wchar_t *s)
+ size_t my_wstrlen(const wchar_t *s)
{
const wchar_t *p = s;
```

Tom 修改过的文件，他先提交这些更改后，他就可以推送了。

```
[tom@CentOS src]$ git commit -a -m 'Resolved conflict'
[master 6b1ac36] Resolved conflict

[tom@CentOS src]$ git pull origin wchar_support.
```

Tom 已经解决冲突，现在推送操作将成功。

Git 不同的平台 - Git教程

GNU/ Linux 和 Mac OS使用换行符（LF）或新行作为行结束字符，而Windows 使用换行和回车（LFCR）的组合来表示行结束字符。

为了避免不必要的提交，因为这些行结束的差异，Git客户端配置写在同一行结束 Git 仓库。

对于Windows系统中，我们可以配置的Git客户端换行符转换为CRLF格式，同时检查了，并把它们转换回LF格式提交操作过程中。下面设置将不可少。

```
[tom@CentOS project]$ git config --global core.autocrlf true
```

对于GNU/ Linux或Mac OS，我们可以配置 Git 客户端CRLF 到 LF换行符转换，同时进行检出（checkout ）操作。

```
[tom@CentOS project]$ git config --global core.autocrlf input
```

GitHub 在线存储库 - Git教程

GitHub是使用Git的版本控制系统是一个基于网络的托管服务的软件开发项目。它也有其标准的GUI应用程序可供下载（在Windows，Mac，GNU/Linux）的直接从服务的网站。但在这个环节中，我们将只能看到CLI部分。

创建GitHub的资料库

去到 github.com. 如果您已经GitHub的帐户，然后使用该帐户登录，或创建新的。从 github.com 网站按照以下步骤来创建新的存储库。

推送操作

Tom 决定使用GitHub上服务器。要开始新的项目，他将创建一个新的目录和一个文件里面。

```
[tom@CentOS]$ mkdir github_repo

[tom@CentOS]$ cd github_repo/

[tom@CentOS]$ vi hello.c

[tom@CentOS]$ make hello
cc hello.c -o hello

[tom@CentOS]$ ./hello
```

上面的命令会产生以下结果。

```
Hello, World !!!
```

在验证自己的代码后，他在初始化目录用git init命令在本地提交他的变化。

```
[tom@CentOS]$ git init
Initialized empty Git repository in /home/tom/github_repo/.git/

[tom@CentOS]$ git status -s
?? hello
?? hello.c

[tom@CentOS]$ git add hello.c

[tom@CentOS]$ git status -s
A hello.c
?? hello

[tom@CentOS]$ git commit -m 'Initial commit'
```

之后，他增加了GitHub的版本库URL作为一个远程的起源，并推他到远程仓库。

> 注：我们已经讨论了所有这些步骤在第4章下创建裸库部分。

```
[tom@CentOS]$ git remote add origin https://github.com/kangralkar/testing_repo.git

[tom@CentOS]$ git push -u origin master
```

推送操作会询问 GitHub 的用户名和密码。验证成功后，操作会成功。

上面的命令会产生以下结果。

```
Username for 'https://github.com': kangralkar
Password for 'https://kangralkar@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 214 bytes, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/kangralkar/test_repo.git
 * [new branch]      master -> master
 Branch master set up to track remote branch master from origin.
```

从现在开始，Tom 可以在 GitHub 库做任何更改。他可以使用本章讨论的所有命令在 GitHub 的仓库中。

Pull 操作

Tom 成功地把他所有的变化GitHub的库。现在其他开发人员可以查看这些更改进行克隆操作或更新他们的本地资源库。

Jerry 在他的home目录和克隆的GitHub库使用git clone命令创建新的目录。

```
[jerry@CentOS]$ pwd
/home/jerry

[jerry@CentOS]$ mkdir jerry_repo

[jerry@CentOS]$ git clone https://github.com/kangralkar/test_repo.git
```

上面的命令会产生以下结果。

```
Cloning into 'test_repo'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0)
Unpacking objects: 100% (3/3), done.
```

他验证通过执行ls命令的目录内容。

```
[jerry@CentOS]$ ls
test_repo

[jerry@CentOS]$ ls test_repo/
hello.c
```

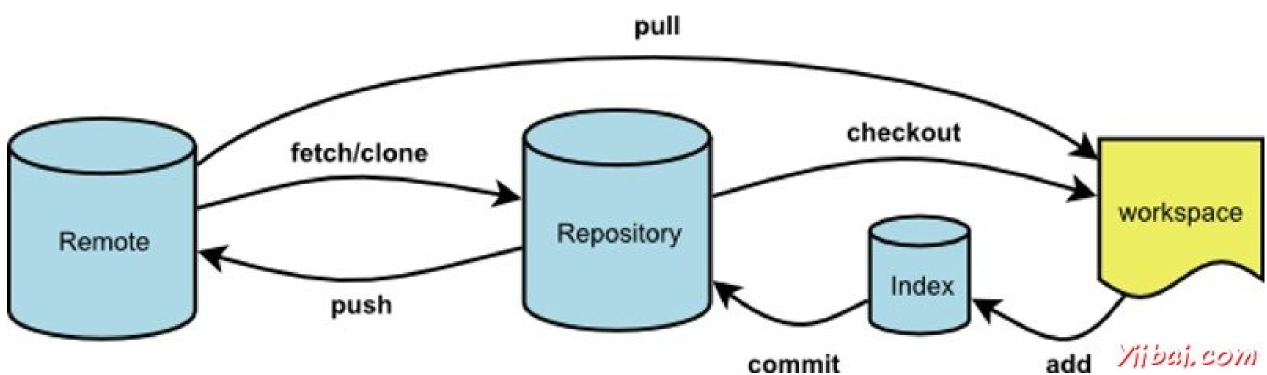
Git远程操作详解 - Git教程

Git是目前最流行的版本管理系统，学会Git几乎成了开发者的必备技能。

Git有很多优势，其中之一就是远程操作非常简便。本文详细介绍5个Git命令，它们的概念和用法，理解了这些内容，你就会完全掌握Git远程操作。

- git clone
- git remote
- git fetch
- git pull
- git push

本文针对初级用户，从最简单的讲起，但是需要读者对Git的基本用法有所了解。同时，本文覆盖了上面5个命令的几乎所有的常用用法，所以对于熟练用户也有参考价值。



Linux 基础

Linux 简介

Linux内核最初只是由芬兰人李纳斯·托瓦兹（Linus Torvalds）在赫尔辛基大学上学时出于个人爱好而编写的。

Linux是一套免费使用和自由传播的类Unix操作系统，是一个基于POSIX和UNIX的多用户、多任务、支持多线程和多CPU的操作系统。

Linux能运行主要的UNIX工具软件、应用程序和网络协议。它支持32位和64位硬件。Linux继承了Unix以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。

Linux的发行版

Linux的发行版说简单点就是将Linux内核与应用软件做一个打包。

目前市面上较知名的发行版有：Ubuntu、RedHat、CentOS、Debain、Fedora、SuSE、OpenSUSE、TurboLinux、BluePoint、RedFlag、Xterm、SlackWare等。

Linux应用领域

今天各种场合都有使用各种Linux发行版，从嵌入式设备到超级计算机，并且在服务器领域确定了地位，通常服务器使用LAMP（Linux + Apache + MySQL + PHP）或LNMP（Linux + Nginx+ MySQL + PHP）组合。

目前Linux不仅在家庭与企业中使用，并且在政府中也很受欢迎。

- 巴西联邦政府由于支持Linux而世界闻名。
- 有新闻报道俄罗斯军队自己制造的Linux发布版的，做为G.H.ost项目已经取得成果.
- 印度的Kerala联邦计划在向全联邦的高中推广使用Linux。
- 中华人民共和国为取得技术独立，在龙芯过程中排他性地使用Linux。
- 在西班牙的一些地区开发了自己的Linux发布版，并且在政府与教育领域广泛使用，如Extremadura地区的gnuLinEx和Andalusia地区的Guadalinex。
- 葡萄牙同样使用自己的Linux发布版Caixa Mágica，用于Magalh?es笔记本电脑和e-escola政府软件。
- 法国和德国同样开始逐步采用Linux。

Linux vs Window

目前国内Linux更多的是应用于服务器上，而桌面操作系统更多使用的是Window。主要区别如下：

比较	Windows	Linux
界面	界面统一，外壳程序固定所有Windows程序菜单几乎一致，快捷键也几乎相同	图形界面风格依发布版不同而不同，可能互不兼容。GNU/Linux的终端机是从UNIX传承下来，基本命令和操作方法也几乎一致。
驱动程序	驱动程序丰富，版本更新频繁。默认安装程序里面一般包含有该版本发布时流行的硬件驱动程序，之后所出的新硬件驱动依赖于硬件厂商提供。对于一些老硬件，如果没有了原配的驱动有时很难支持。另外，有时硬件厂商未提供所需版本的Windows下的驱动，也会比较头痛。	由志愿者开发，由Linux核心开发小组发布，很多硬件厂商基于版权考虑并未提供驱动程序，尽管多数无需手动安装，但是涉及安装则相对复杂，使得新用户面对驱动程序问题（是否存在和安装方法）会一筹莫展。但是在开源开发模式下，许多老硬件尽管在Windows下很难支持的也容易找到驱动。HP、Intel、AMD等硬件厂商逐步不同程度支持开源驱动，问题正在得到缓解。
使用	使用比较简单，容易入门。图形化界面对没有计算机背景知识的用户使用十分有利。	图形界面使用简单，容易入门。文字界面，需要学习才能掌握。
学习	系统构造复杂、变化频繁，且知识、技能淘汰快，深入学习困难。	系统构造简单、稳定，且知识、技能传承性好，深入学习相对容易。
软件	每一种特定功能可能都需要商业软件的支持，需要购买相应的授权。	大部分软件都可以自由获取，同样功能的软件选择较少。

Linux 安装

本章节我们将为大家介绍Linux的安装。

本章节以 centos6.4 为例。

centos6.4 下载地址：

- 网易镜像：http://mirrors.163.com/centos/6/isos/
- 搜狐镜像：http://mirrors.sohu.com/centos/6/isos/

注：建议安装64位Linux系统。

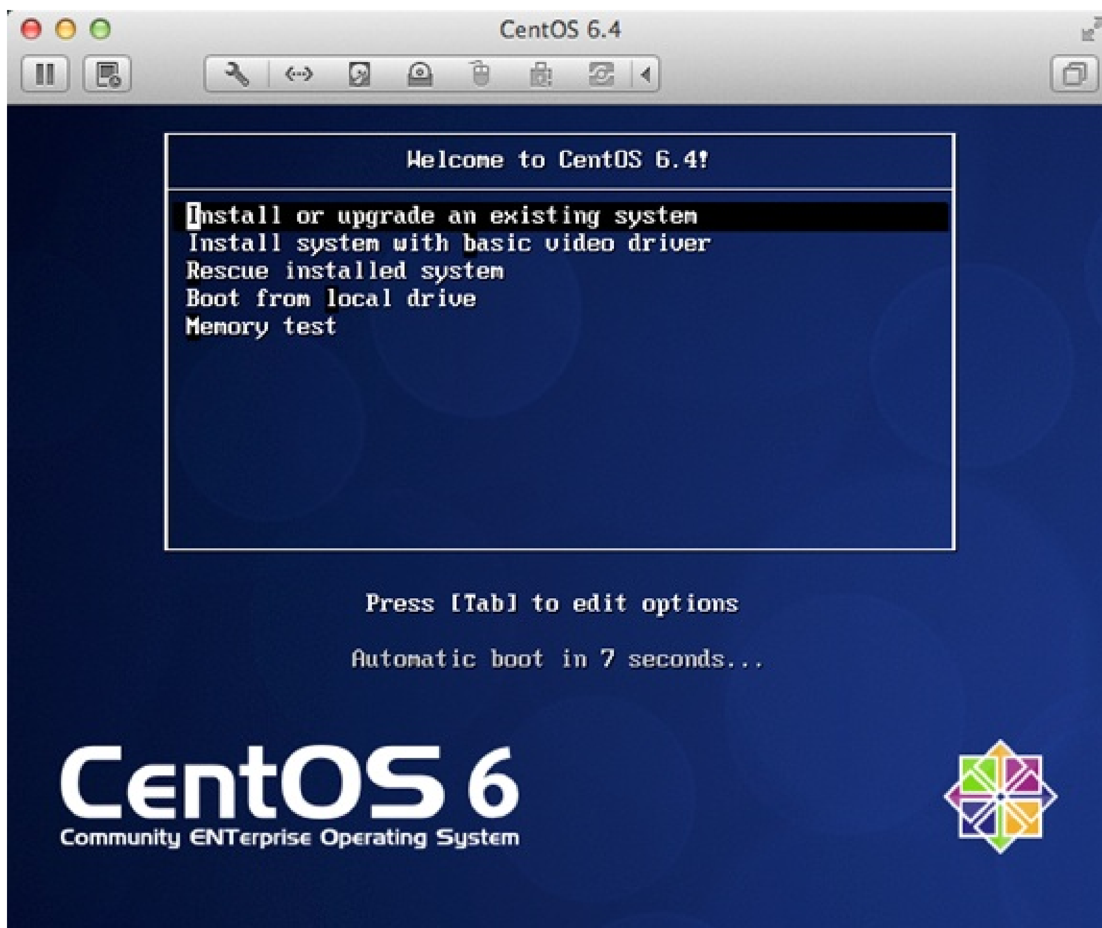
接下来你需要将下载的Linux系统刻录成光盘或U盘。

注：你也可以在Window上安装VMware虚拟机来安装Linux系统。

Linux 安装步骤

1、首先，使用光驱或U盘或你下载的Linux ISO文件进行安装。

界面说明：



Install or upgrade an existing system 安装或升级现有的系统

install system with basic video driver 安装过程中采用基本的显卡驱动

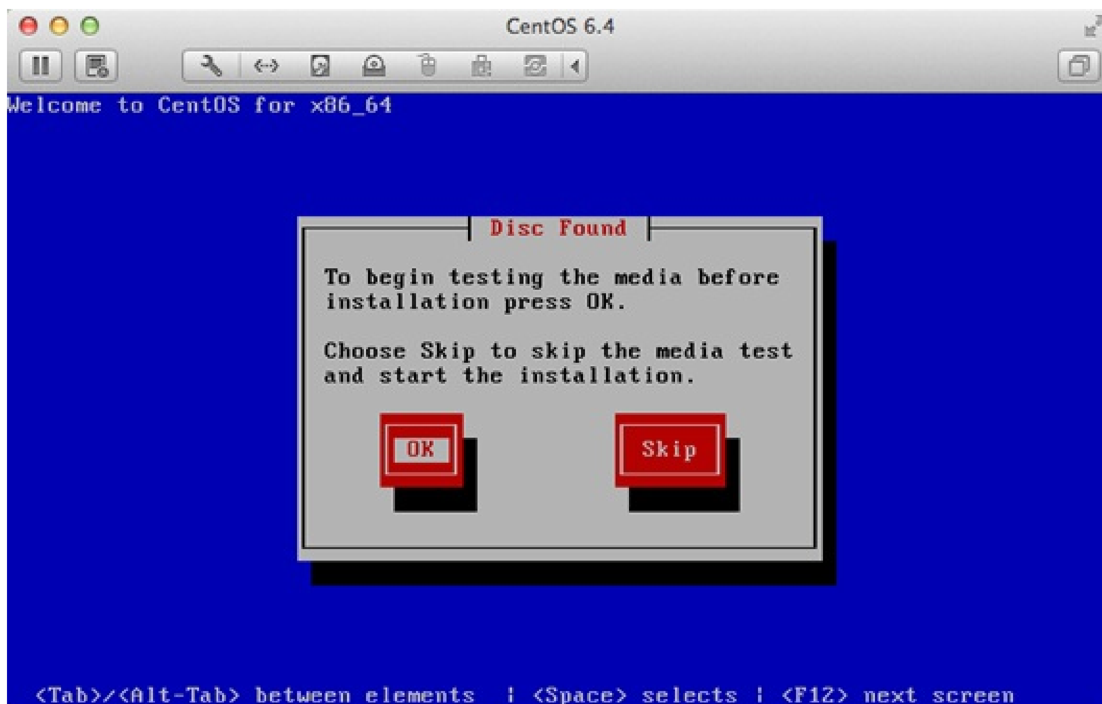
Rescue installed system 进入系统修复模式

Boot from local drive 退出安装从硬盘启动

Memory test 内存检测

注：用联想E49安装时选择第一项安装时会出现屏幕显示异常的问题，后改用第二项安装时就没有出现问题

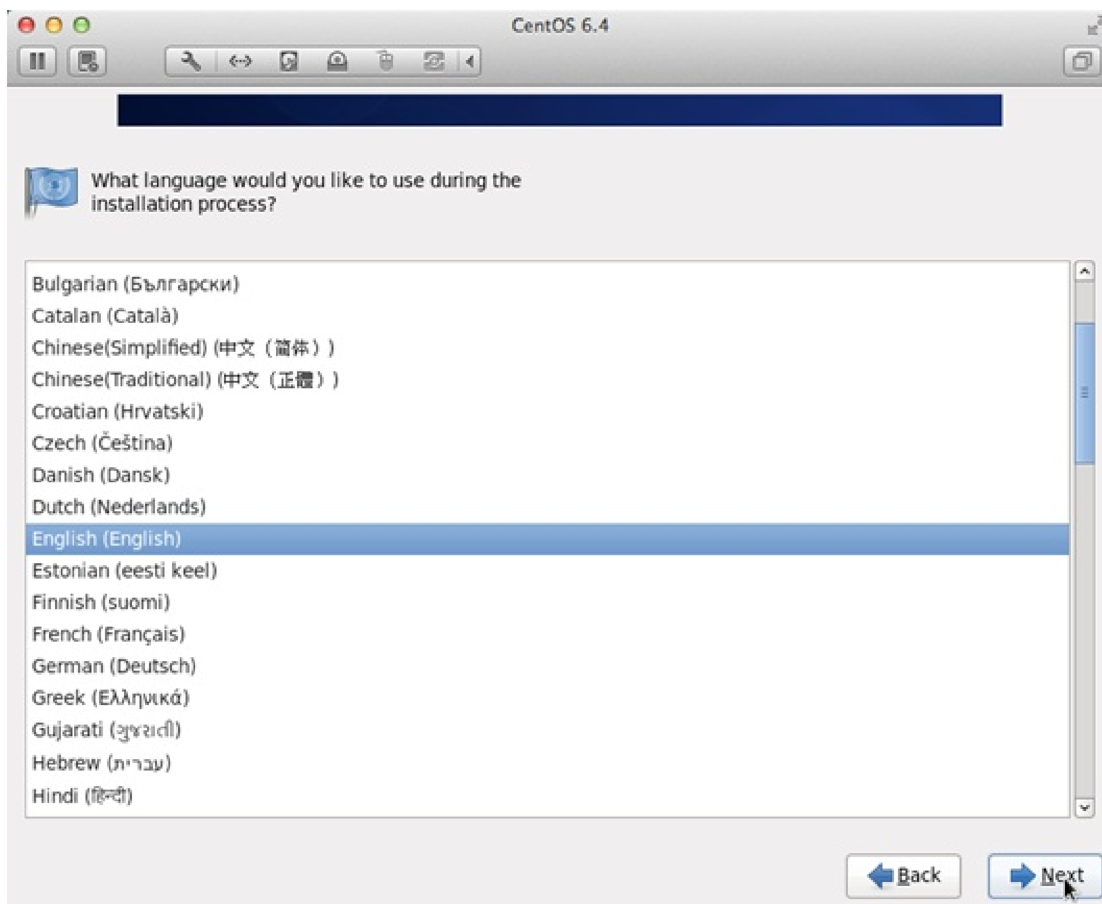
2、介质直接"skip"就可以了



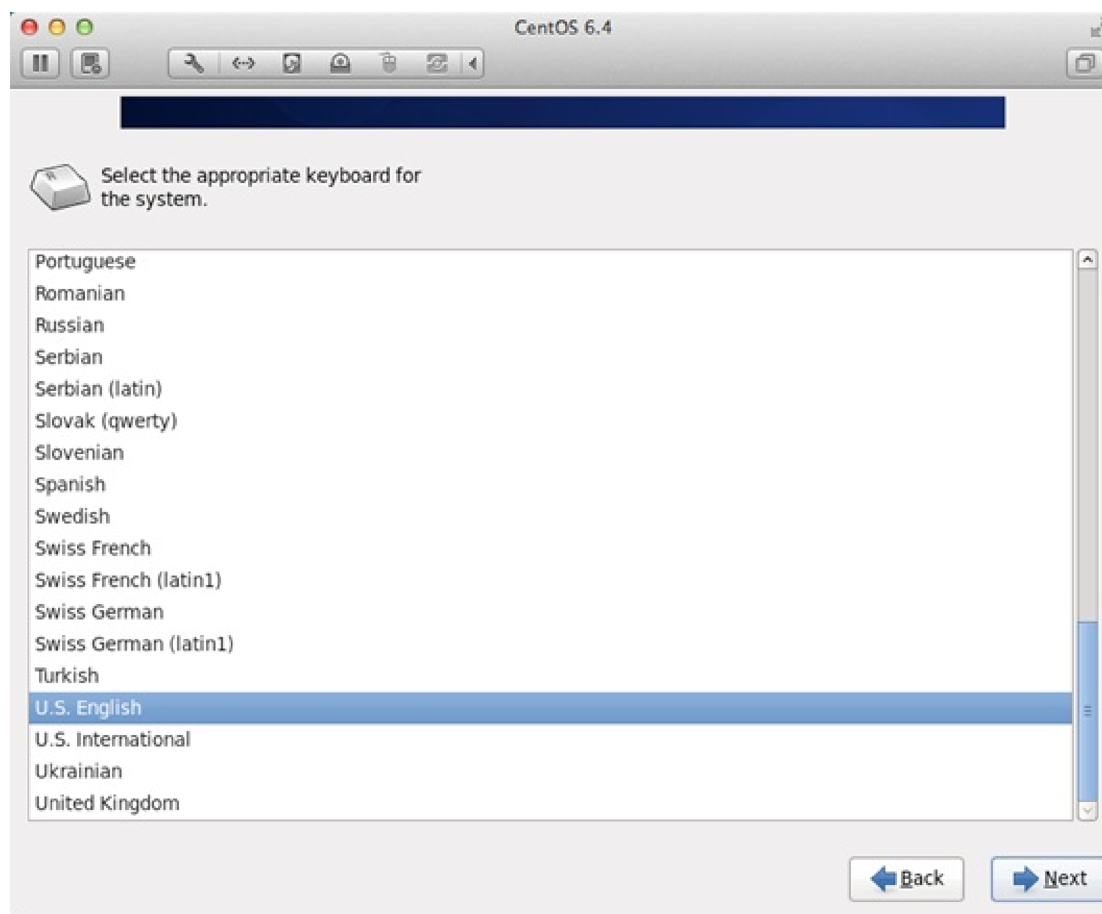
3、出现引导界面，点击"next"



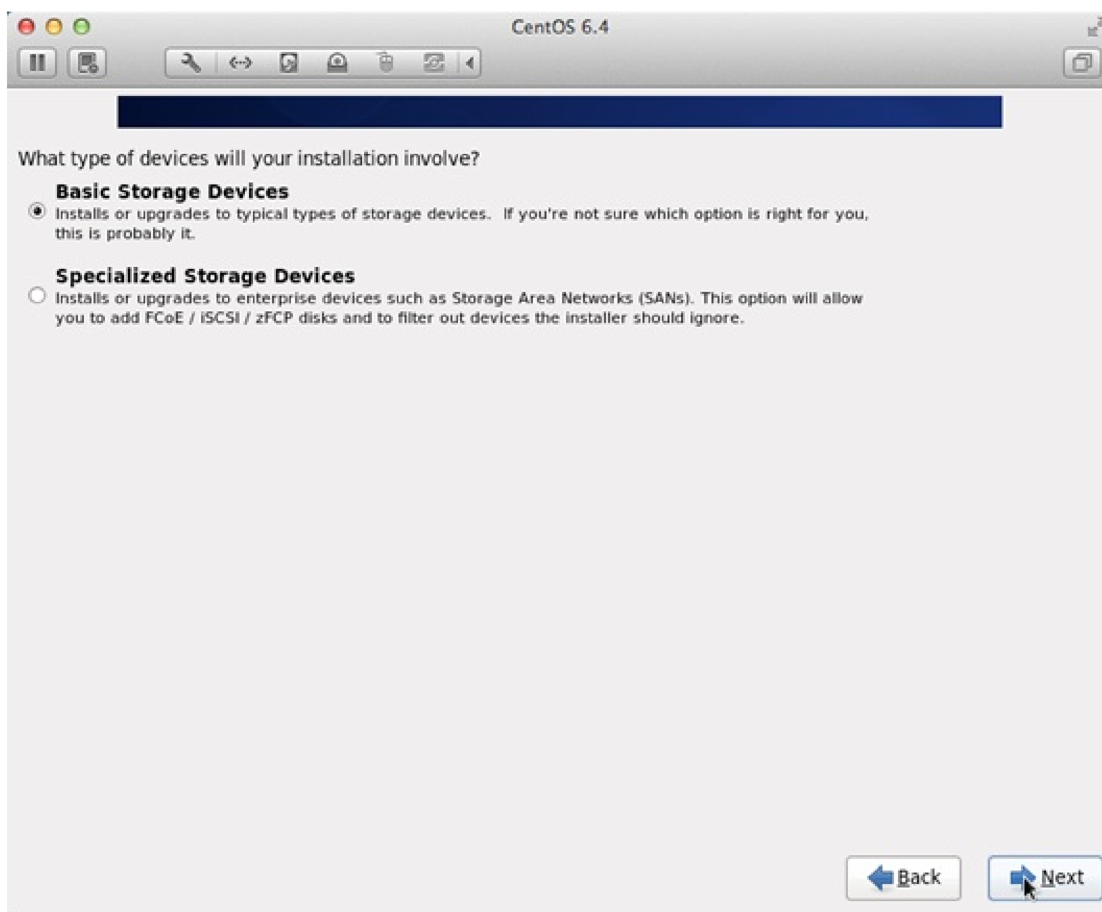
4、选中"English（English）"否则会有部分乱码问题



5、键盘布局选择"U.S.English"



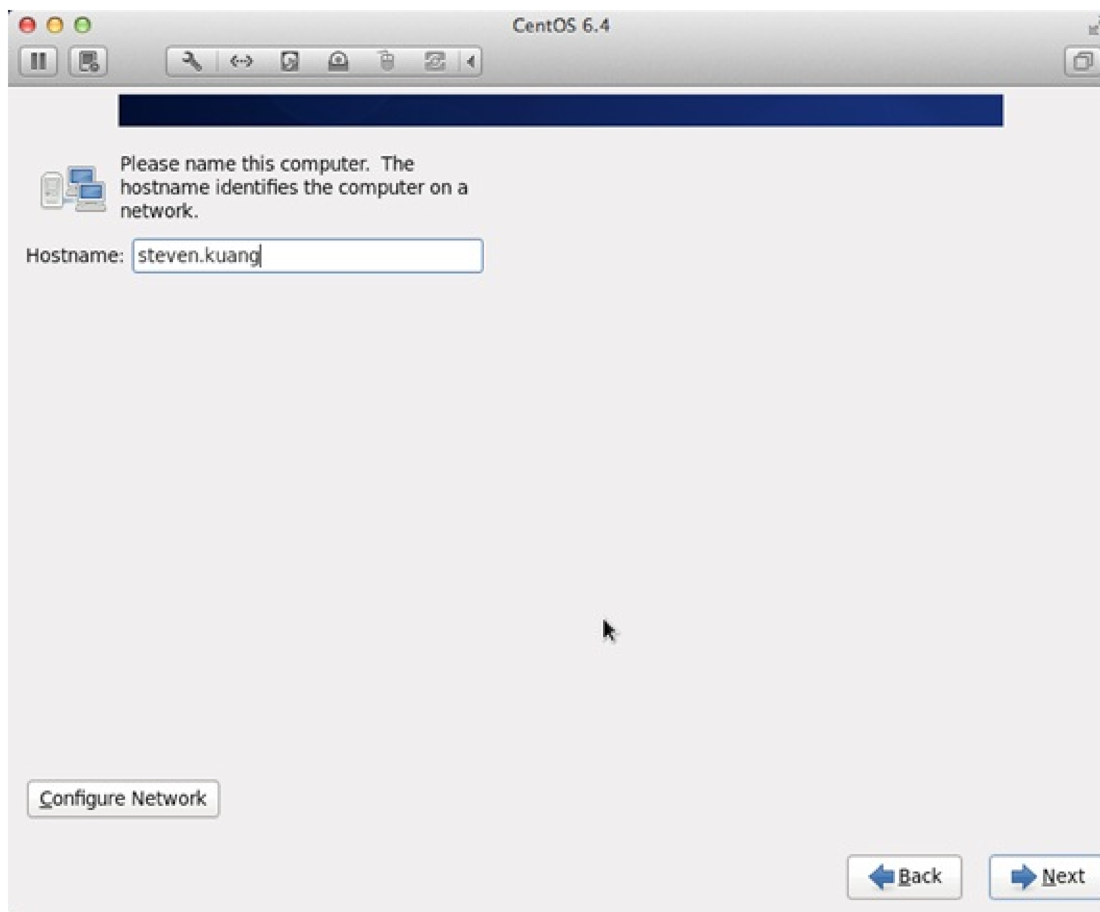
6、选择"Basic Storage Devies"点击"Next"



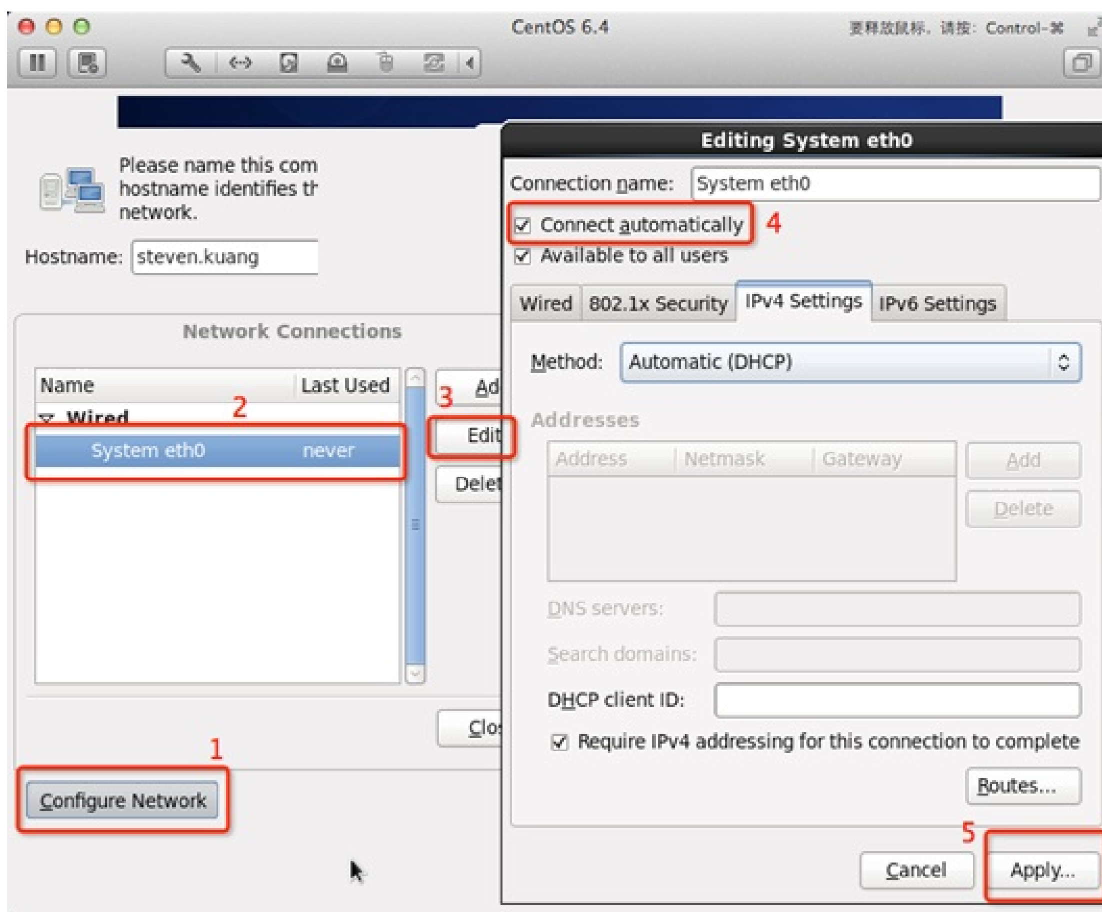
7、询问是否忽略所有数据，新电脑安装系统选择"Yes,discard any data"



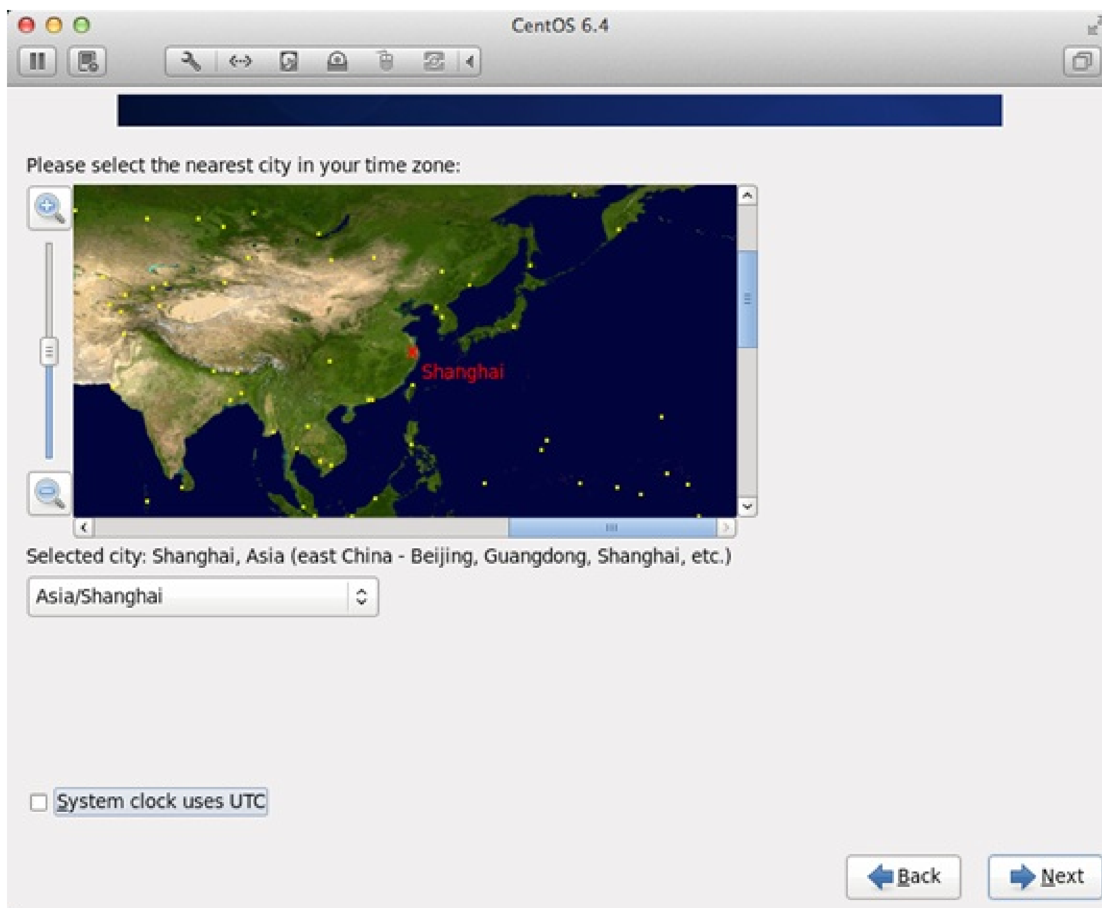
8、Hostname填写格式"英文名.姓"



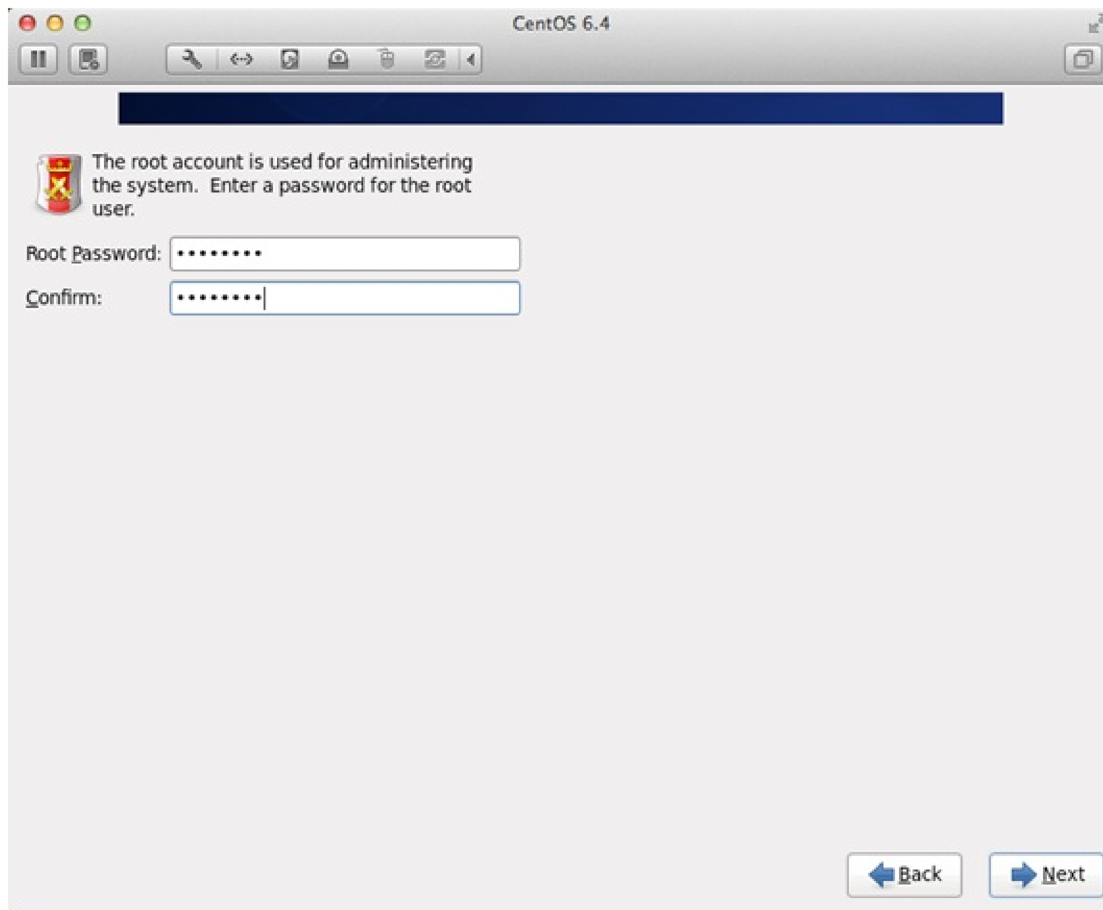
9、网络设置安装图示顺序点击就可以了



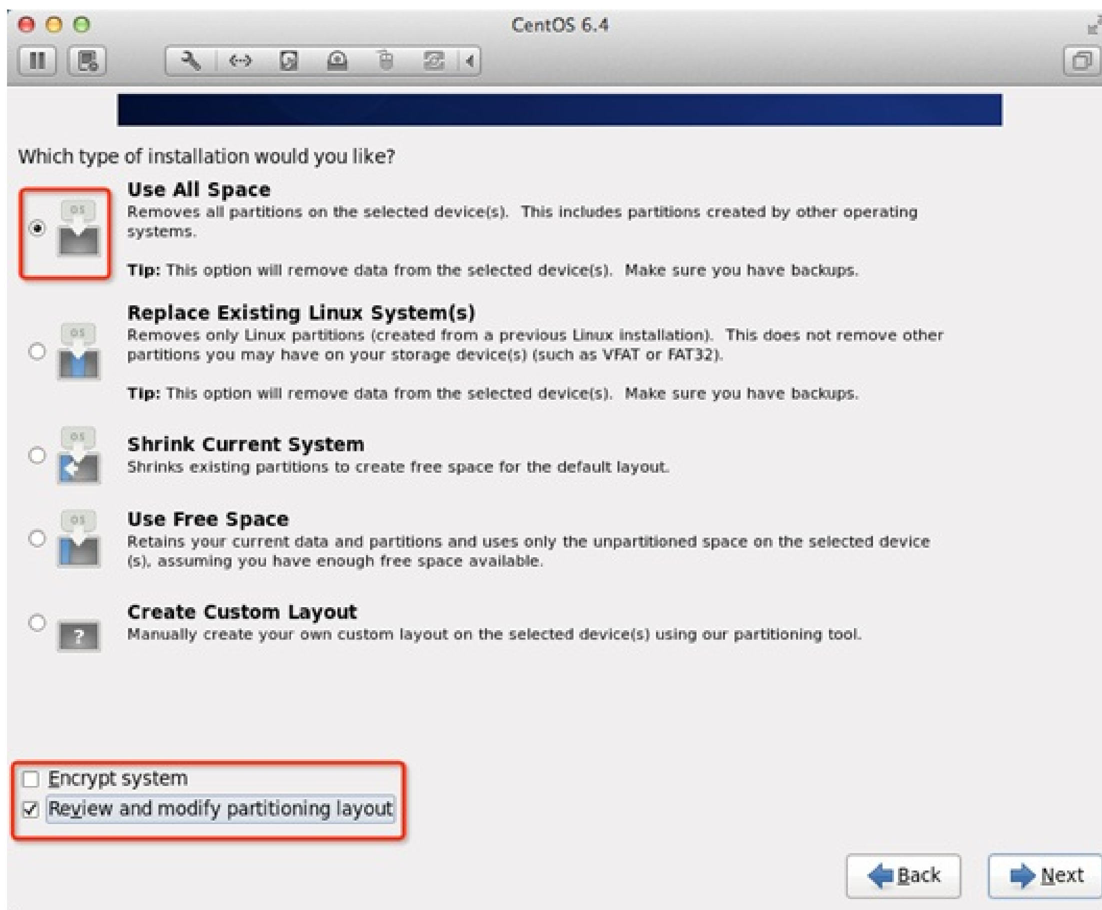
10、 时区可以在地图上点击，选择"shanghai"并取消System clock uses UTC前面的对勾



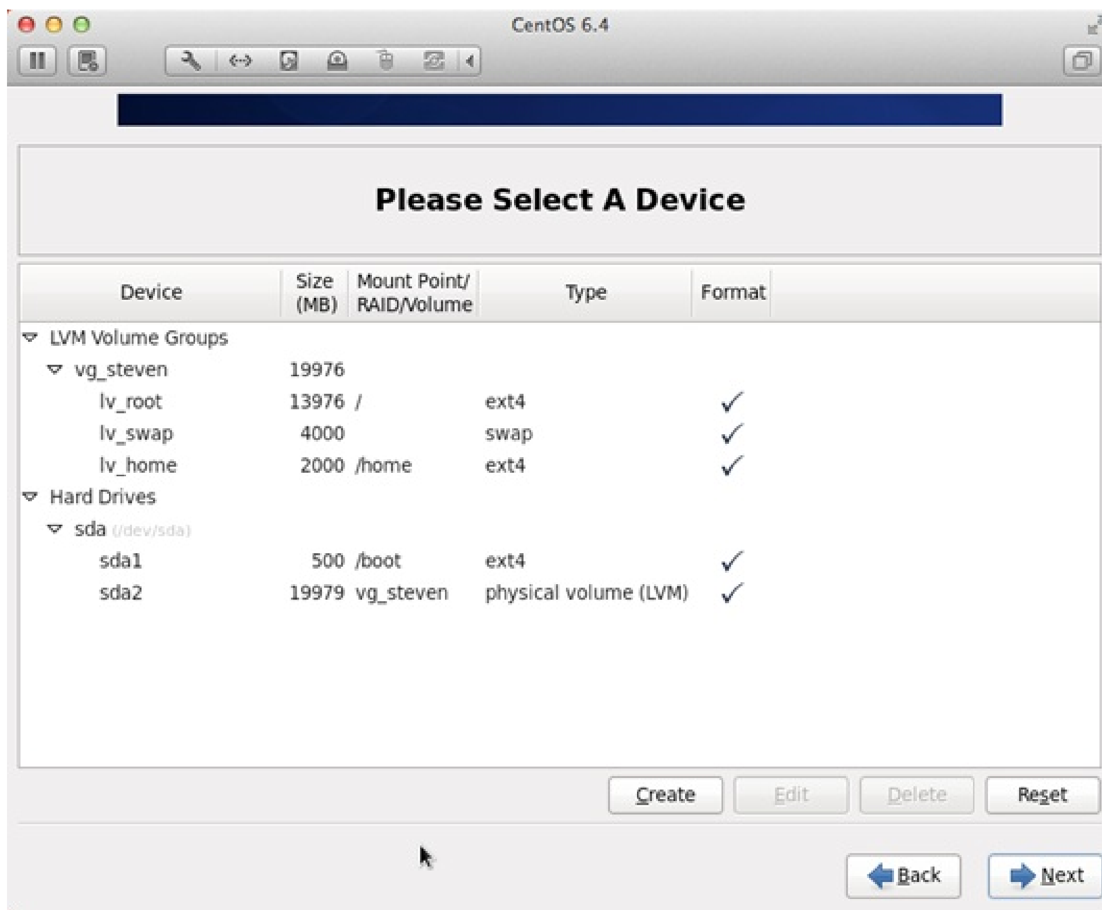
11、设置root的密码



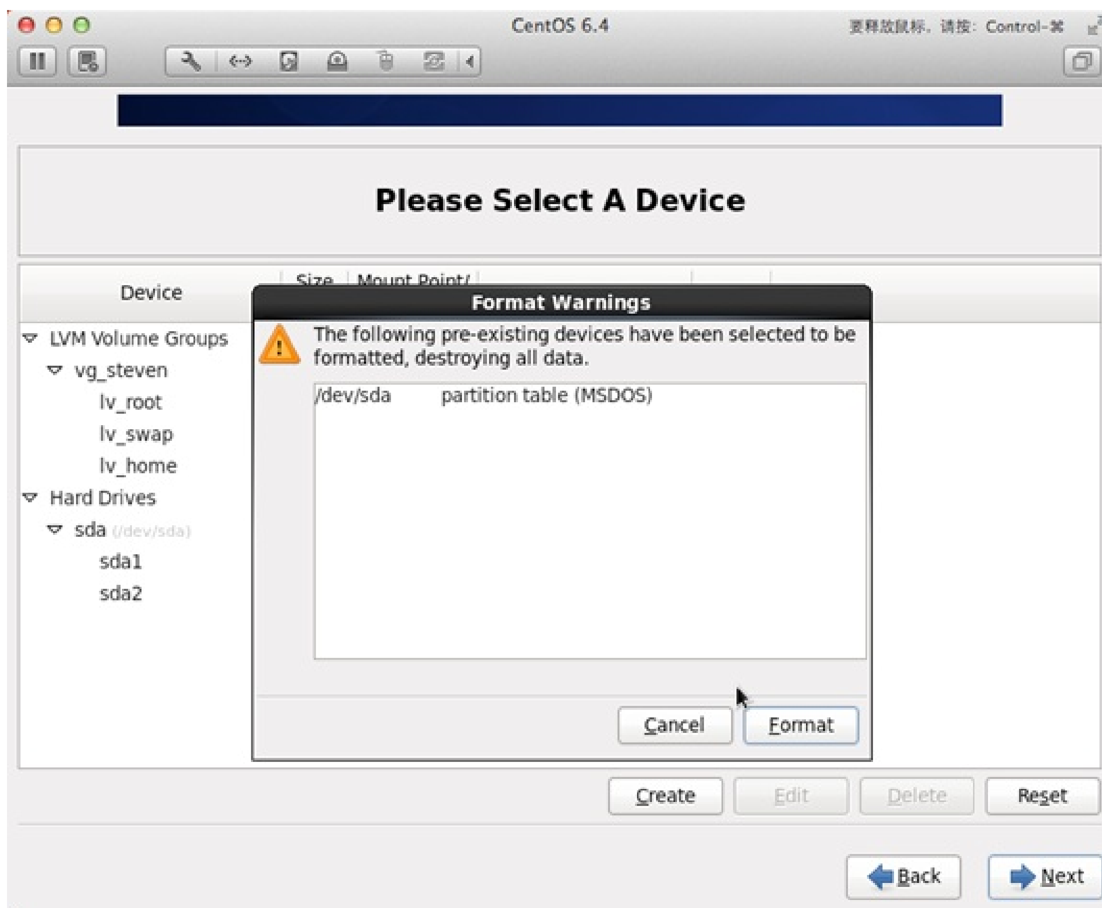
12、硬盘分区，一定要按照图示点选



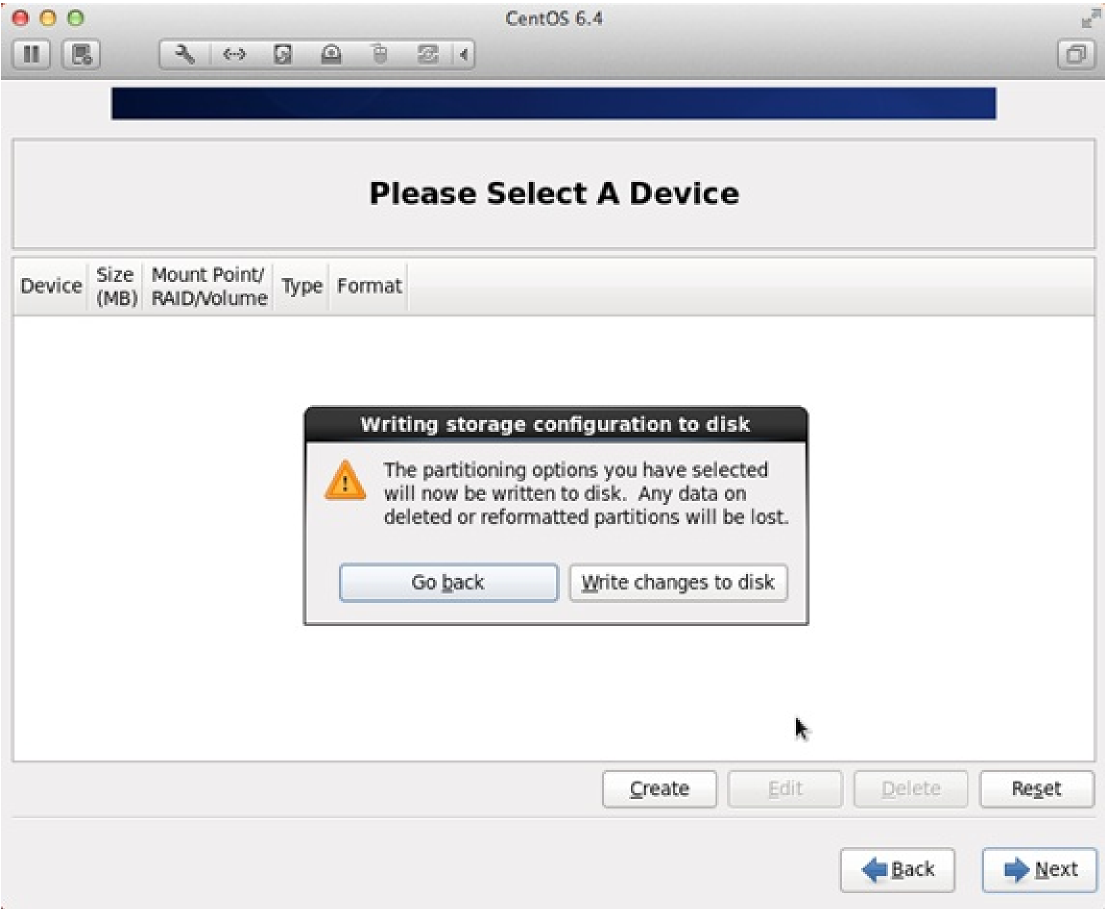
13、调整分区，必须要有/home这个分区，如果没有这个分区，安装部分软件会出现不能安装的问题



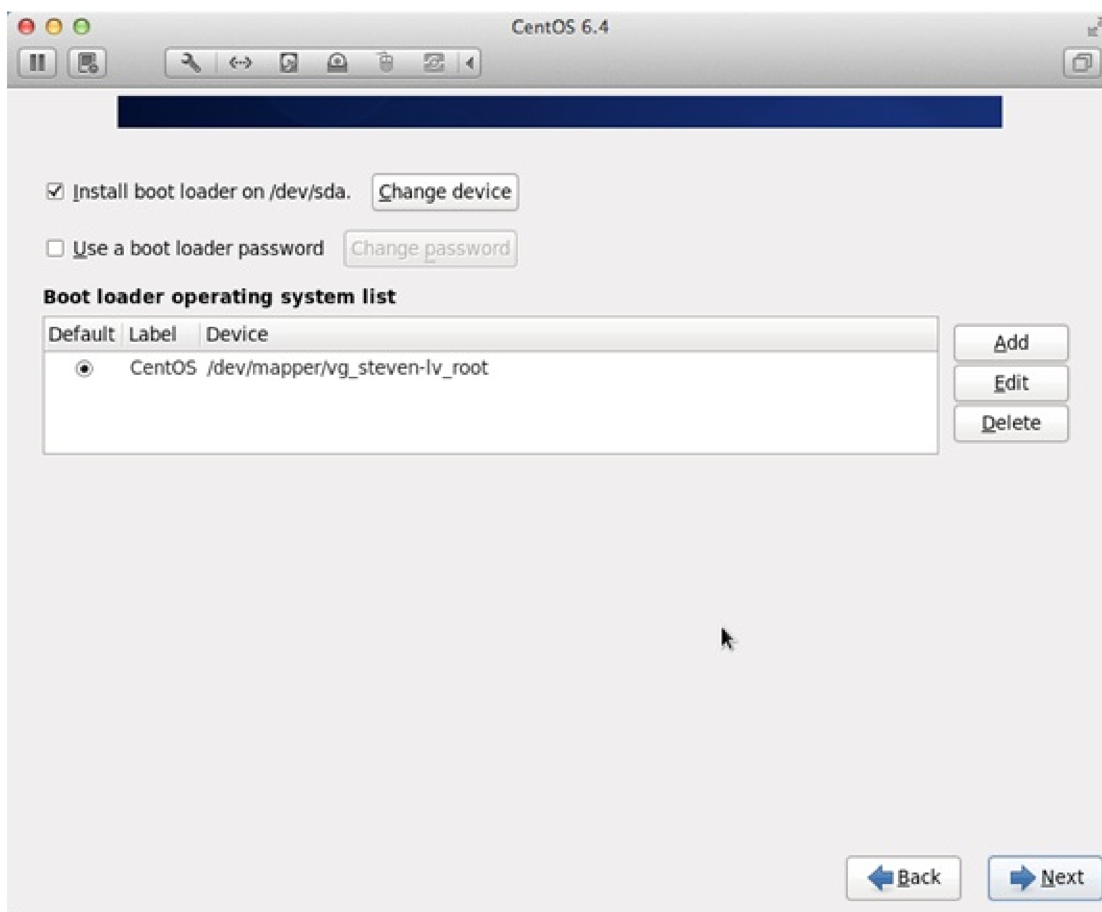
14、询问是否格式化分区



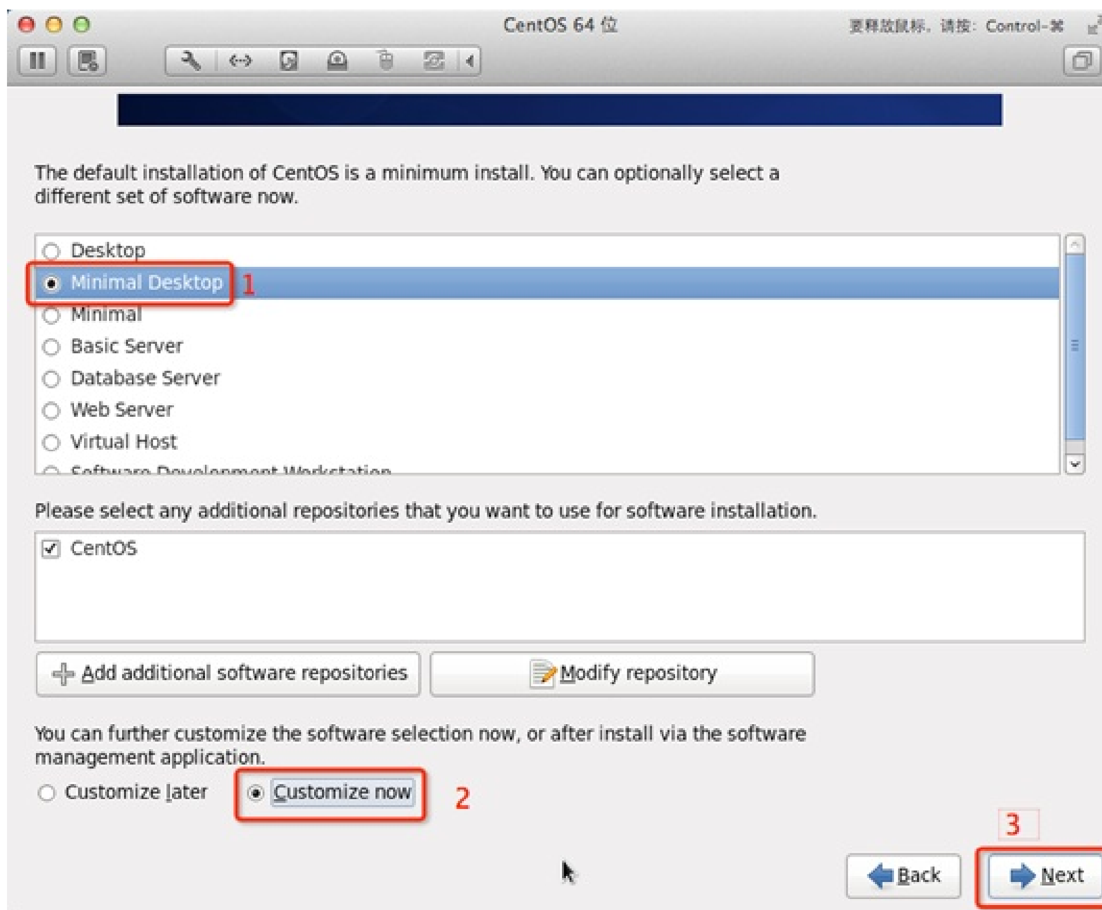
15、将更改写入到硬盘



16、引导程序安装位置



17、最重要的一步，也是本教程最关机的一步，也是其他教程没有提及的一步，按图示顺序点击



18、取消以下内容的所有选项

Applications

Base System

Servers

并对Desktops进行如下设置

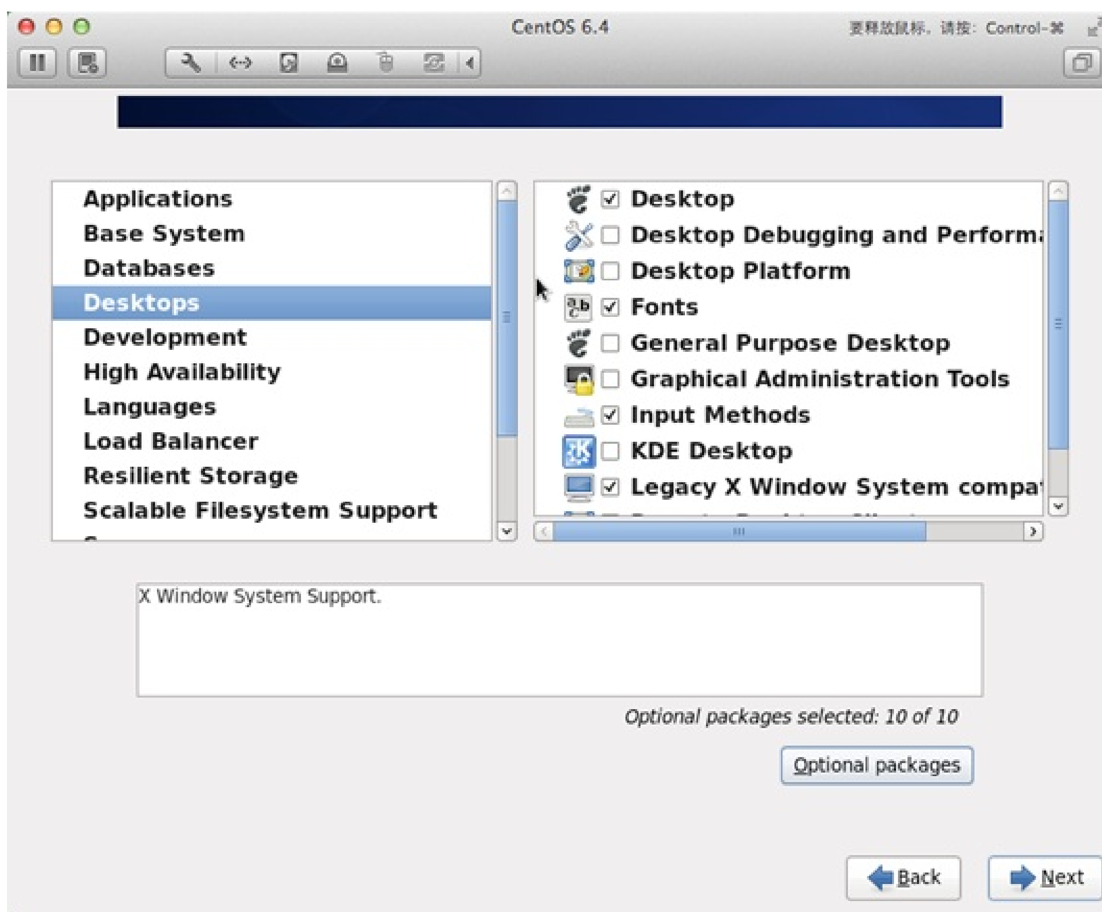
即取消如下选项：

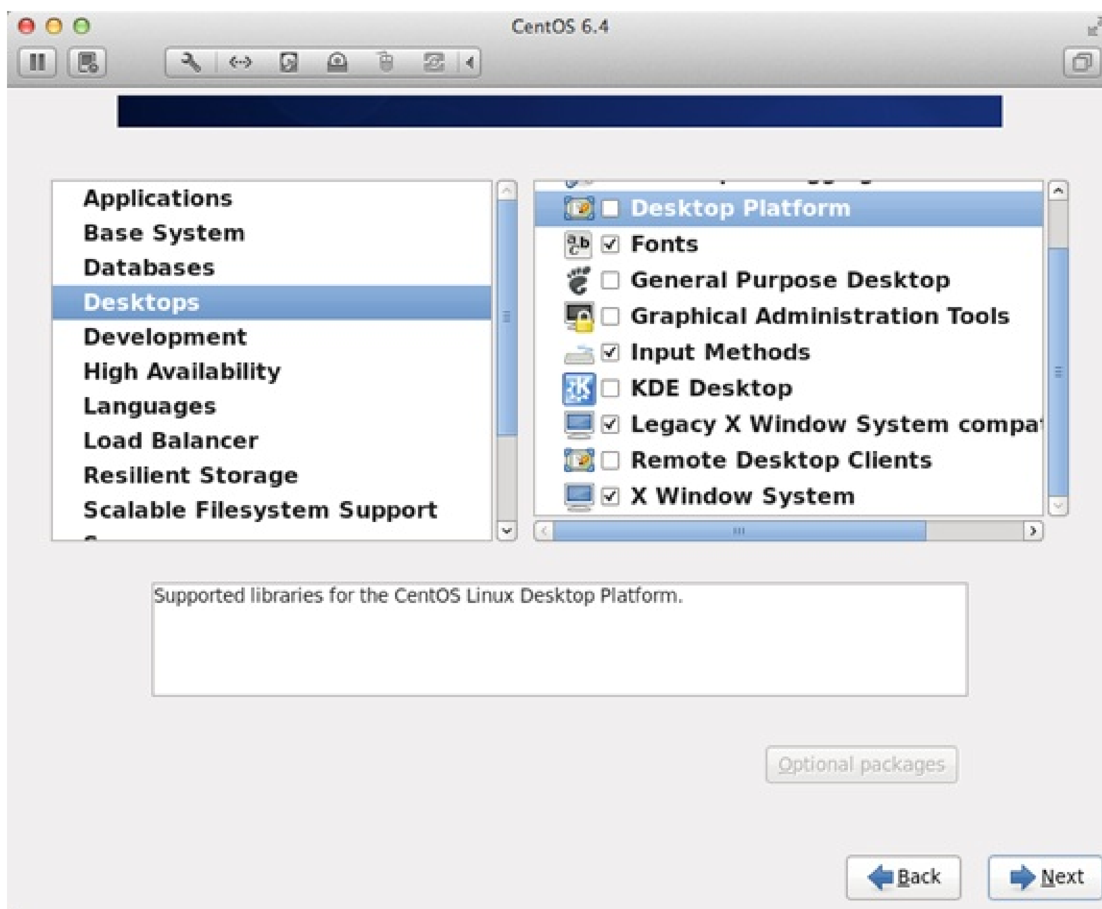
Desktop Debugging and Performance Tools

Desktop Platform

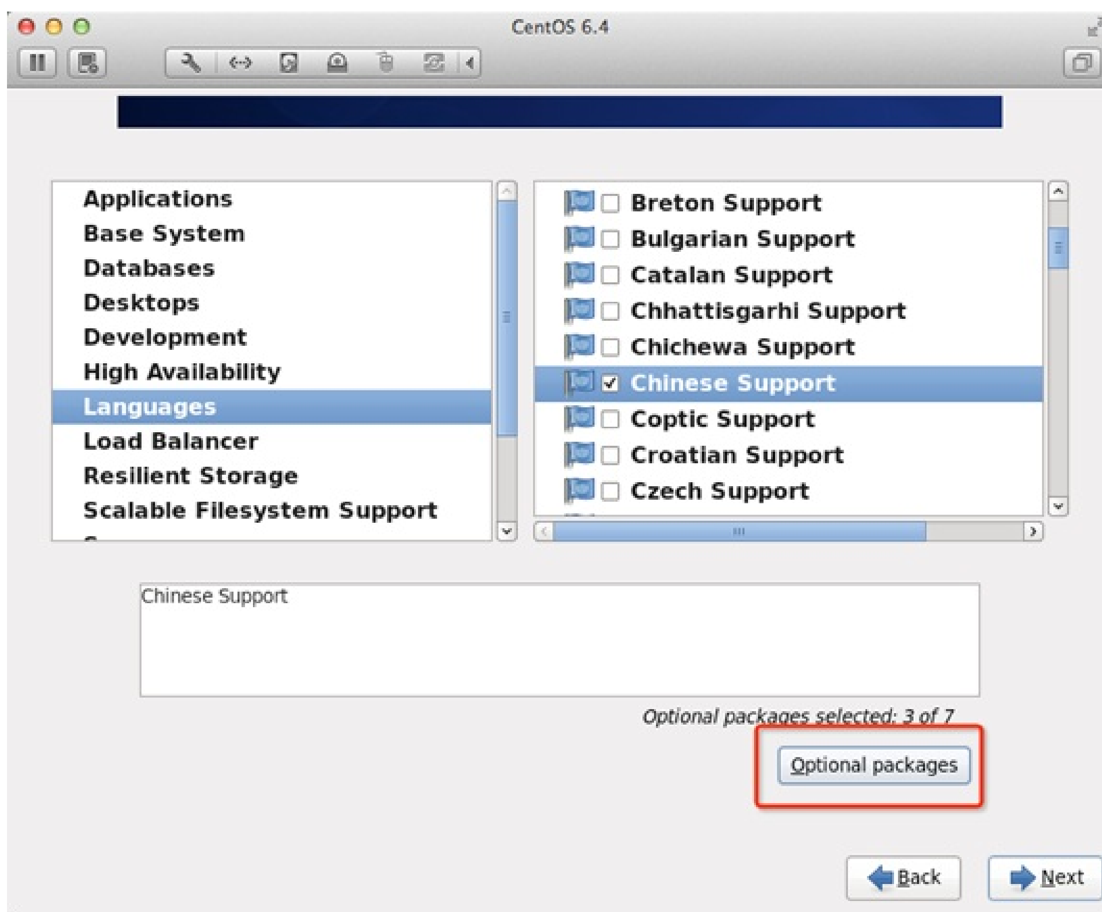
Remote Desktop Clients

Input Methods中仅保留ibus-pinyin-1.3.8-1.el6.x86_64,其他的全部取消**

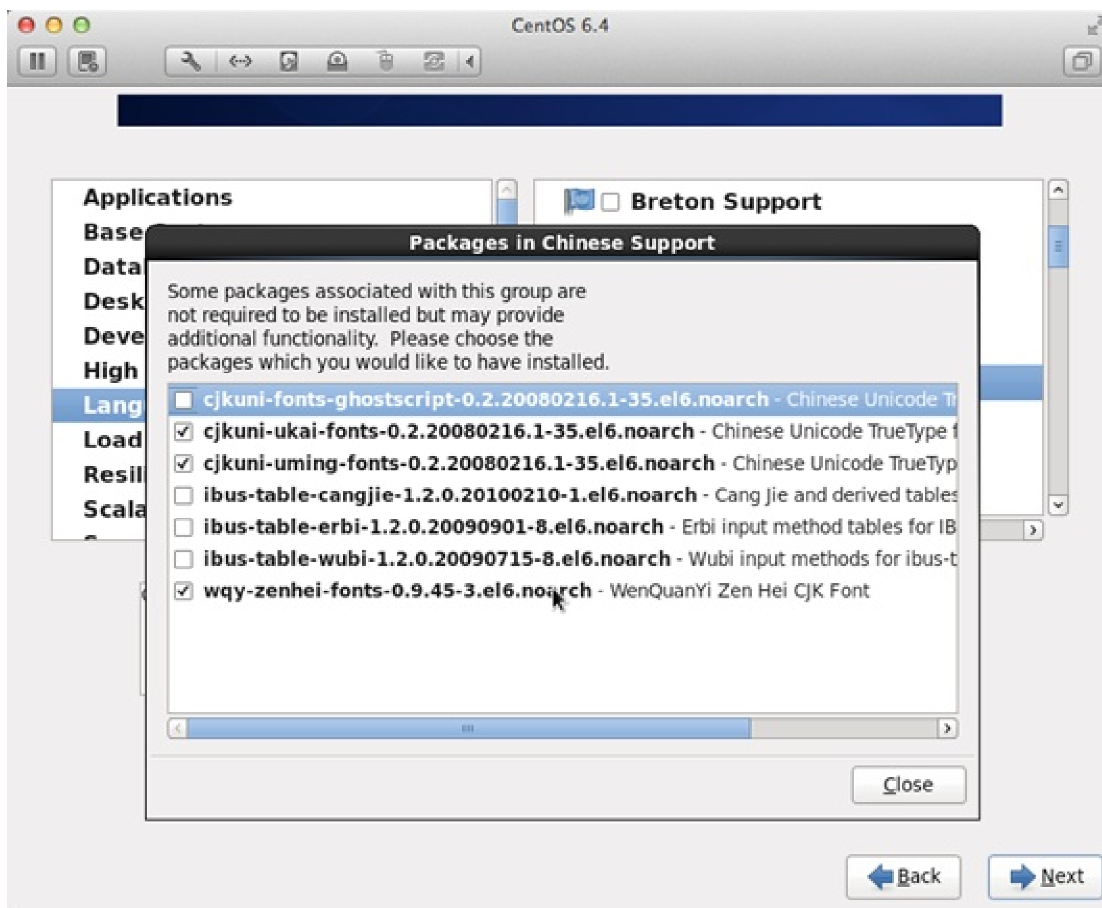




19、选中Languages，并选中右侧的Chinese Support然后点击红色区域



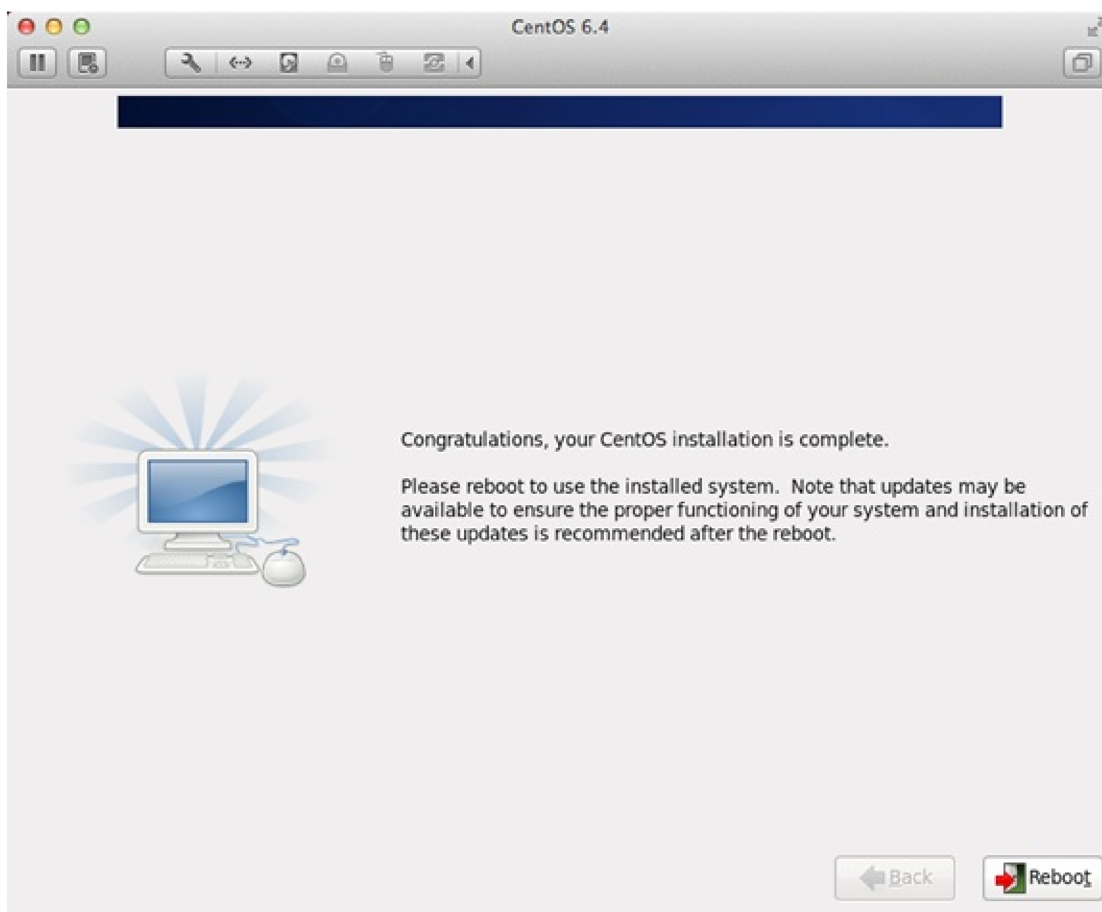
20、调整完成后如下图所示



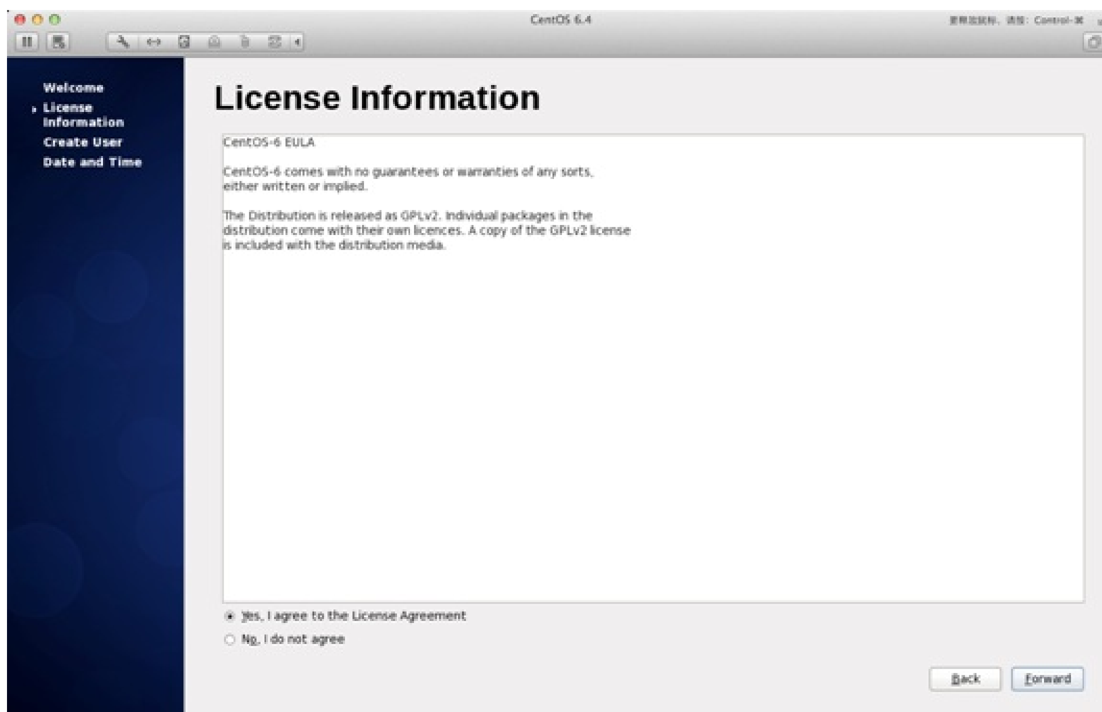
21、至此，一个最精简的桌面环境就设置完成了，



22、安装完成，重启



23、重启之后，的License Information



24、Create User

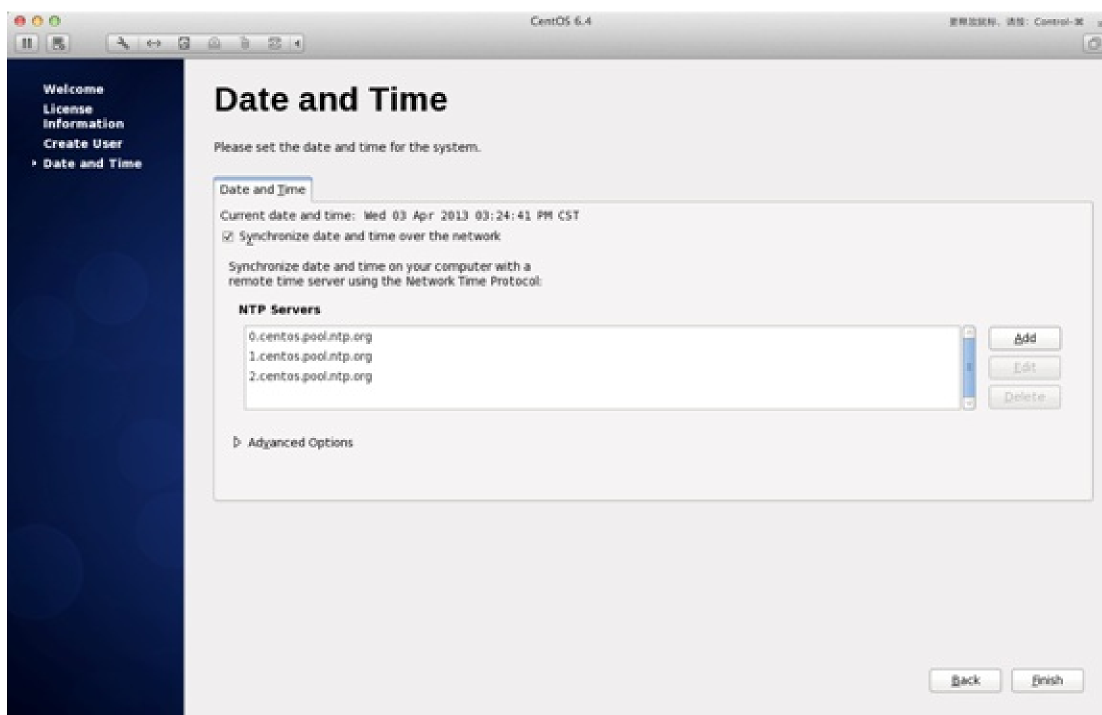
Username：填写您的英文名（不带.姓）

Full Name：填写您的英文名.姓（首字母大写）



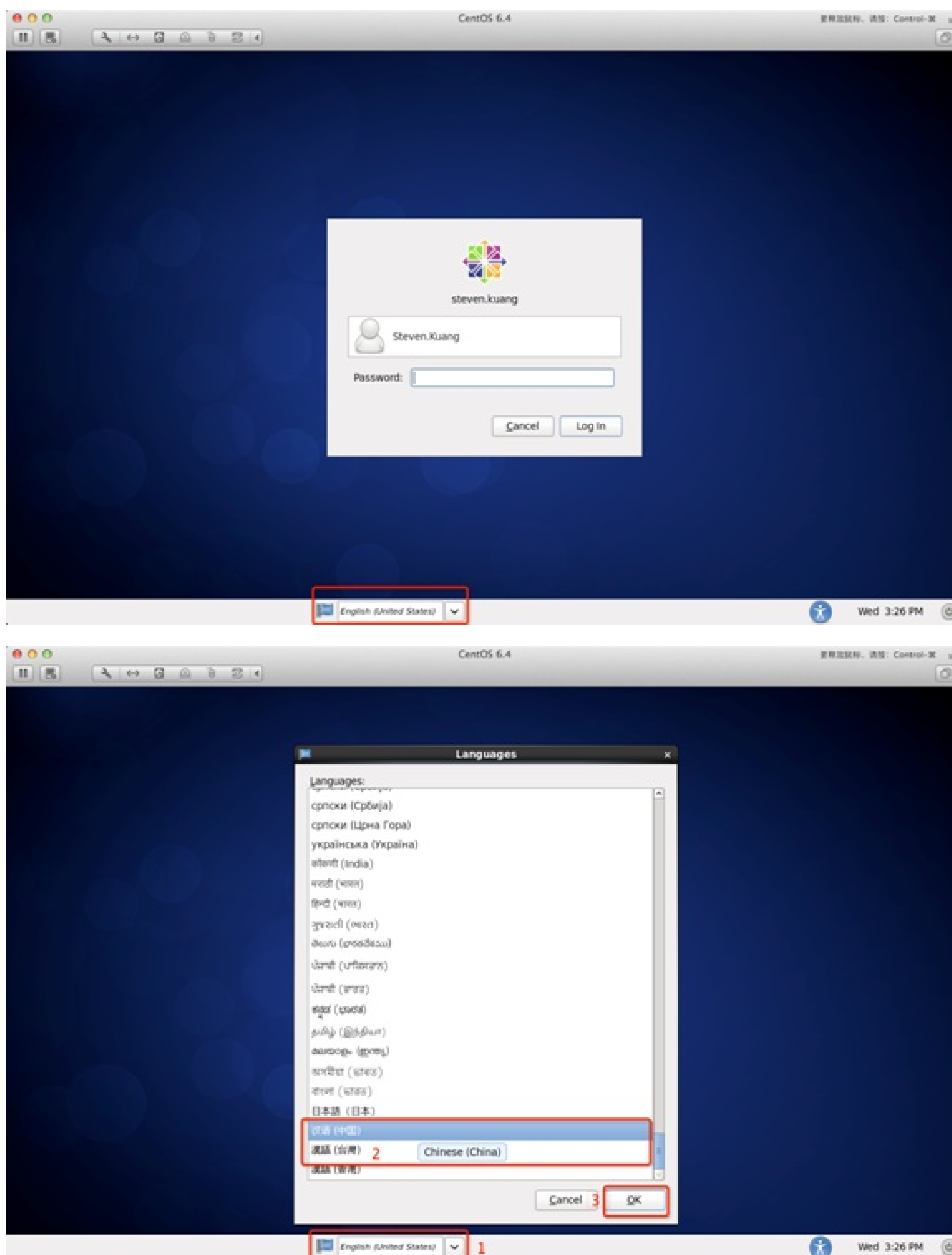
25、"Date and Time" 选中 "Synchronize data and time over the network"

Finsh之后系统将重启



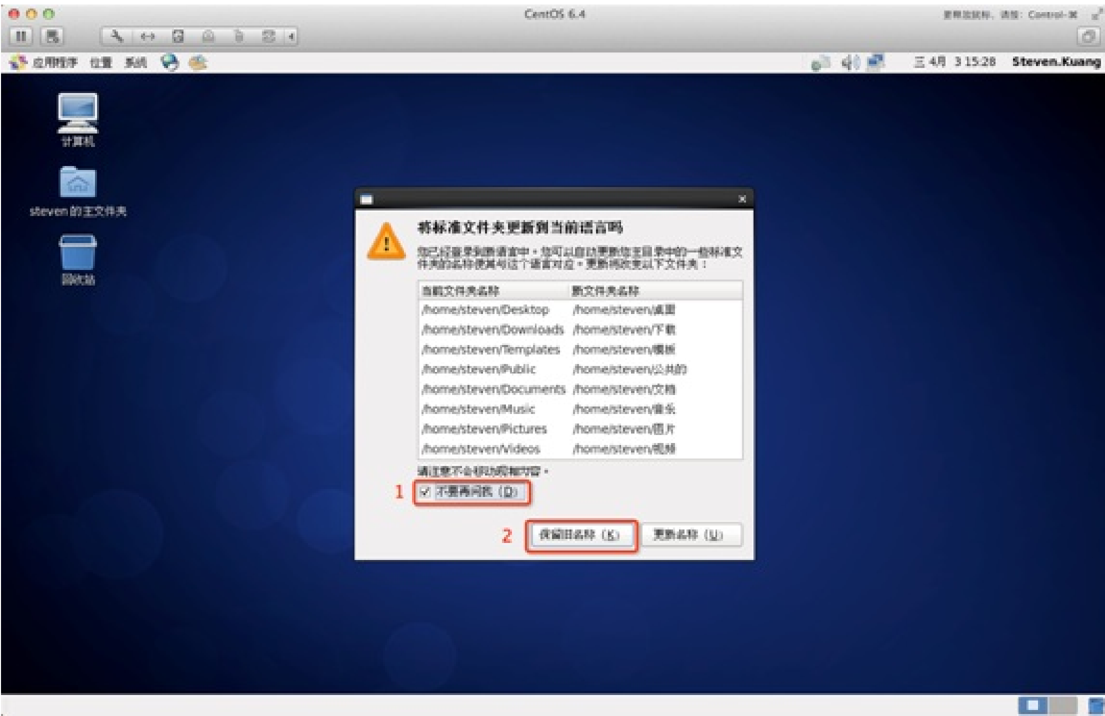
26、第一次登录，登录前不要做任何更改，这个很重要！！！登录之后紧接着退出

第二次登录，选择语言，在红色区域选择下拉小三角，选other，选中"汉语（中国）"



27、登录之后，请一定按照如下顺序点击！

至此，CentOS安装完成，如有其他问题，请随时与我联系！！


CentOS 6.4
应用程序 位置 系统
Steven.Kuang
计算机
steven 的主文件夹
回收站
将标准文件夹更新到当前语言吗
当前文件夹名称
新文件夹名称
/home/steven/Desktop
/home/steven/桌面
/home/steven/Downloads
/home/steven/下载
/home/steven/Templates
/home/steven/模板
/home/steven/Public
/home/steven/公共的
/home/steven/Documents
/home/steven/文档
/home/steven/Music
/home/steven/音乐
/home/steven/Pictures
/home/steven/图片
/home/steven/Videos
/home/steven/视频
1
不要再问我（D）
2
保留旧名称（K）
更新名称（U）

Linux 系统启动过程

linux启动时我们会看到许多启动信息。

Linux系统的启动过程并不是大家想象中的那么复杂，其过程可以分为5个阶段：

- 内核的引导。
- 运行init。
- 系统初始化。
- 建立终端 。
- 用户登录系统。

内核引导

当计算机打开电源后，首先是BIOS开机自检，按照BIOS中设置的启动设备（通常是硬盘）来启动。

操作系统接管硬件以后，首先读入 /boot 目录下的内核文件。



运行init

init 进程是系统所有进程的起点，你可以把它比拟成系统所有进程的老祖宗，没有这个进程，系统中任何进程都不会启动。

init 程序首先是需要读取配置文件 /etc/inittab。



运行级别

许多程序需要开机启动。它们在Windows叫做"服务"（service），在Linux就叫做"守护进程"（daemon）。

init进程的一大任务，就是去运行这些开机启动的程序。

但是，不同的场合需要启动不同的程序，比如用作服务器时，需要启动Apache，用作桌面就不需要。

Linux允许为不同的场合，分配不同的开机启动程序，这就叫做"运行级别"（runlevel）。也就是说，启动时根据"运行级别"，确定要运行哪些程序。



Linux系统有7个运行级别(runlevel)：

- 运行级别0：系统停机状态，系统默认运行级别不能设为0，否则不能正常启动
- 运行级别1：单用户工作状态，root权限，用于系统维护，禁止远程登陆
- 运行级别2：多用户状态(没有NFS)
- 运行级别3：完全的多用户状态(有NFS)，登陆后进入控制台命令行模式
- 运行级别4：系统未使用，保留
- 运行级别5：X11控制台，登陆后进入图形GUI模式
- 运行级别6：系统正常关闭并重启，默认运行级别不能设为6，否则不能正常启动

系统初始化

在init的配置文件中有这么一行： si::sysinit:/etc/rc.d/rc.sysinit　它调用执行了/etc/rc.d/rc.sysinit，而rc.sysinit是一个bash shell的脚本，它主要是完成一些系统初始化的工作，rc.sysinit是每一个运行级别都要首先运行的重要脚本。

它主要完成的工作有：激活交换分区，检查磁盘，加载硬件模块以及其它一些需要优先执行任务。

```
l5:5:wait:/etc/rc.d/rc 5
```

这一行表示以5为参数运行/etc/rc.d/rc，/etc/rc.d/rc是一个Shell脚本，它接受5作为参数，去执行/etc/rc.d/rc5.d/目录下的所有的rc启动脚本，/etc/rc.d/rc5.d/目录中的这些启动脚本实际上都是一些连接文件，而不是真正的rc启动脚本，真正的rc启动脚本实际上都是放在/etc/rc.d/init.d/目录下。

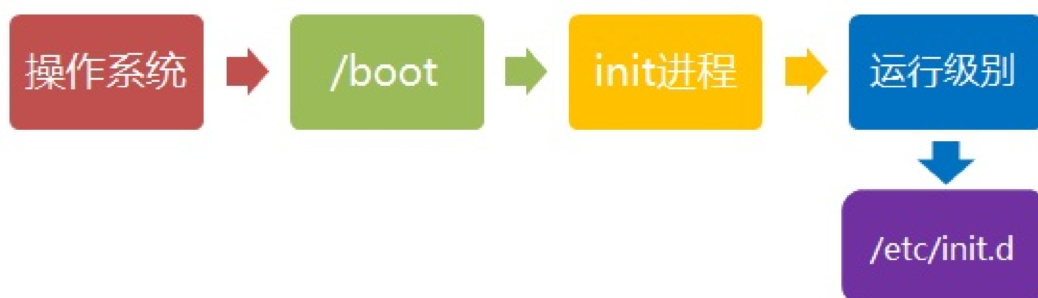
而这些rc启动脚本有着类似的用法，它们一般能接受start、stop、restart、status等参数。

/etc/rc.d/rc5.d/中的rc启动脚本通常是K或S开头的连接文件，对于以以S开头的启动脚本，将以start参数来运行。

而如果发现存在相应的脚本也存在K打头的连接，而且已经处于运行态了(以/var/lock/subsys/下的文件作为标志)，则将首先以stop为参数停止这些已经启动了的守护进程，然后再重新运行。

这样做是为了保证是当init改变运行级别时，所有相关的守护进程都将重启。

至于在每个运行级中将运行哪些守护进程，用户可以通过chkconfig或setup中的"System Services"来自行设定。



建立终端

rc执行完毕后，返回init。这时基本系统环境已经设置好了，各种守护进程也已经启动了。

init接下来会打开6个终端，以便用户登录系统。在inittab中的以下6行就是定义了6个终端：

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

从上面可以看出在2、3、4、5的运行级别中都将以respawn方式运行mingetty程序，mingetty程序能打开终端、设置模式。

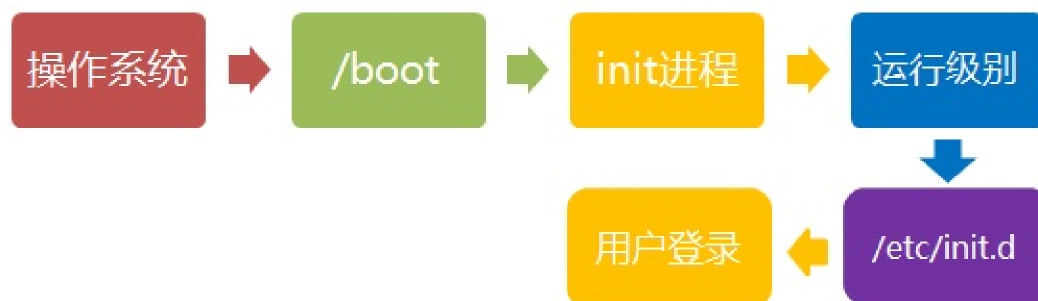
同时它会显示一个文本登录界面，这个界面就是我们经常看到的登录界面，在这个登录界面中会提示用户输入用户名，而用户输入的用户将作为参数传给login程序来验证用户的身份。

用户登录系统

一般来说，用户的登录方式有三种：

- （1）命令行登录

- （2）ssh登录
- （3）图形界面登录



对于运行级别为5的图形方式用户来说，他们的登录是通过一个图形化的登录界面。登录成功后可以直接进入KDE、Gnome等窗口管理器。

而本文主要讲的还是文本方式登录的情况：当我们看到mingetty的登录界面时，我们就可以输入用户名和密码来登录系统了。

Linux的账号验证程序是login，login会接收mingetty传来的用户名作为用户名参数。

然后login会对用户名进行分析：如果用户名不是root，且存在/etc/nologin文件，login将输出nologin文件的内容，然后退出。

这通常用来系统维护时防止非root用户登录。只有/etc/securetty中登记了的终端才允许root用户登录，如果不存在这个文件，则root可以在任何终端上登录。

/etc/usertty文件用于对用户作出附加访问限制，如果不存在这个文件，则没有其他限制。

图形模式与文字模式的切换方式

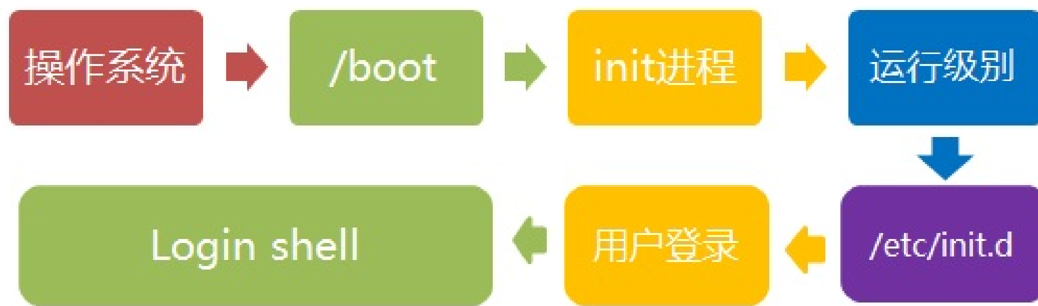
Linux预设提供了六个命令窗口终端机让我们来登录。

默认我们登录的就是第一个窗口，也就是tty1，这个六个窗口分别为tty1,tty2 … tty6，你可以按下Ctrl + Alt + F1 ~ F6 来切换它们。

如果你安装了图形界面，默认情况下是进入图形界面的，此时你就可以按Ctrl + Alt + F1 ~ F6来进入其中一个命令窗口界面。

当你进入命令窗口界面后再返回图形界面只要按下Ctrl + Alt + F7 就回来了。

如果你用的vmware 虚拟机，命令窗口切换的快捷键为 Alt + Space + F1~F6. 如果你在图形界面下请按Alt + Shift + Ctrl + F1~F6 切换至命令窗口。



Linux 关机

在linux领域内大多用在服务器上，很少遇到关机的操作。毕竟服务器上跑一个服务是永无止境的，除非特殊情况下，不得已才会关机。

正确的关机流程为：sysnc > shutdown > reboot > halt

关机指令为：shutdown， 你可以man shutdown 来看一下帮助文档。

例如你可以运行如下命令关机：

```
sync 将数据由内存同步到硬盘中。
shutdown 关机指令，你可以man shutdown 来看一下帮助文档。例如你可以运行如下命令关机：
shutdown –h 10 ‘This server will shutdown after 10 mins’ 这个命令告诉大家，计算机将在10分钟后关
Shutdown –h now 立马关机
Shutdown –h 20:25 系统会在今天20:25关机
Shutdown –h +10 十分钟后关机
Shutdown –r now 系统立马重启
Shutdown –r +10 系统十分钟后重启
reboot 就是重启，等同于 shutdown –r now
halt 关闭系统，等同于shutdown –h now 和 poweroff
```

最后总结一下，不管是重启系统还是关闭系统，首先要运行sync命令，把内存中的数据写到磁盘中。

关机的命令有 shutdown –h now halt poweroff 和 init 0 , 重启系统的命令有 shutdown –r now reboot init 6.

Linux 系统目录结构

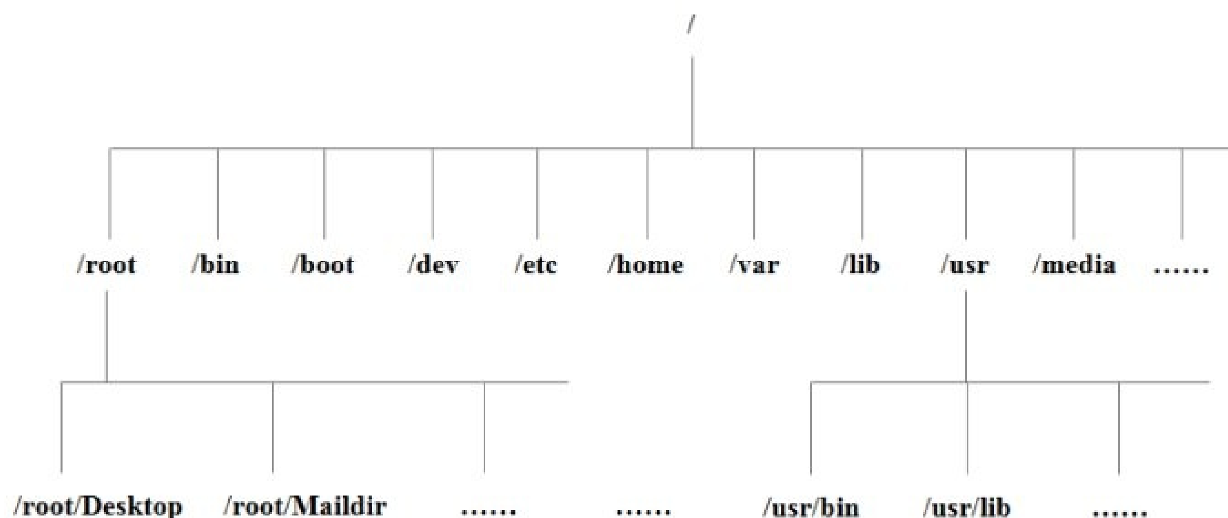
登录系统后，在当前命令窗口下输入命令：

```
ls /
```

你会看到如下图所示:

```
[root@localhost ~]# ls /
bin   dev  home  lost+found  mnt  proc  sbin     srv  tmp  var
boot  etc  lib   media       opt  root  selinux  sys  usr
```

树状目录结构：



以下是对这些目录的解释：

- **/bin**：
  bin是Binary的缩写, 这个目录存放着最经常使用的命令。

- **/boot**：
  这里存放的是启动Linux时使用的一些核心文件，包括一些连接文件以及镜像文件。

- **/dev** ：
  dev是Device(设备)的缩写, 该目录下存放的是Linux的外部设备，在Linux中访问设备的方式和访问文件的方式是相同的。

- **/etc**：
  这个目录用来存放所有的系统管理所需要的配置文件和子目录。

- **/home**：
  用户的主目录，在Linux中，每个用户都有一个自己的目录，一般该目录名是以用户的账号命名的。

- **/lib**：
  这个目录里存放着系统最基本的动态连接共享库，其作用类似于Windows里的DLL文件。几乎所有的应用程序都需要用到这些共享库。

- **/lost+found**：
  这个目录一般情况下是空的，当系统非法关机后，这里就存放了一些文件。

- /media linux系统会自动识别一些设备，例如U盘、光驱等等，当识别后，linux会把识别的设备挂载到这个目录下。

- **/mnt**：
  系统提供该目录是为了让用户临时挂载别的文件系统的，我们可以将光驱挂载在/mnt/上，然后进入该目录就可以查看光驱里的内容了。

- **/opt**：
  这是给主机额外安装软件所摆放的目录。比如你安装一个ORACLE数据库则就可以放到这个目录下。默认是空的。

- **/proc**：
  这个目录是一个虚拟的目录，它是系统内存的映射，我们可以通过直接访问这个目录来获取系统信息。
  这个目录的内容不在硬盘上而是在内存里，我们也可以直接修改里面的某些文件，比如可以通过下面的命令来屏蔽主机的ping命令，使别人无法ping你的机器：

  ```
  echo 1 &gt; /proc/sys/net/ipv4/icmp_echo_ignore_all
  ```

- **/root**：
  该目录为系统管理员，也称作超级权限者的用户主目录。

- **/sbin**：
  s就是Super User的意思，这里存放的是系统管理员使用的系统管理程序。

- **/selinux**：
  这个目录是Redhat/CentOS所特有的目录，Selinux是一个安全机制，类似于windows的防火墙，但是这套机制比较复杂，这个目录就是存放selinux相关的文件的。

- **/srv**：
  该目录存放一些服务启动之后需要提取的数据。

- **/sys**：
  这是linux2.6内核的一个很大的变化。该目录下安装了2.6内核中新出现的一个文件系统 sysfs 。

  sysfs文件系统集成了下面3种文件系统的信息：针对进程信息的proc文件系统、针对设备的devfs文件系统以及针对伪终端的devpts文件系统。

该文件系统是内核设备树的一个直观反映。

当一个内核对象被创建的时候，对应的文件和目录也在内核对象子系统种被创建。

- **/tmp**：
  这个目录是用来存放一些临时文件的。

- **/usr**：
  这是一个非常重要的目录，用户的很多应用程序和文件都放在这个目录下，类似与windows下的program files目录。

- **/usr/bin**：
  系统用户使用的应用程序。

- **/usr/sbin**：
  超级用户使用的比较高级的管理程序和系统守护程序。

- **/usr/src**：内核源代码默认的放置目录。

- **/var**：
  这个目录中存放着在不断扩充着的东西，我们习惯将那些经常被修改的目录放在这个目录下。包括各种日志文件。

在linux系统中，有几个目录是比较重要的，平时需要注意不要误删除或者随意更改内部文件。

/etc： 上边也提到了，这个是系统中的配置文件，如果你更改了该目录下的某个文件可能会导致系统不能启动。

/bin, /sbin, /usr/bin, /usr/sbin: 这是系统预设的执行文件的放置目录，比如 ls 就是在/bin/ls 目录下的。

值得提出的是，/bin, /usr/bin 是给系统用户使用的指令（除root外的通用户），而/sbin, /usr/sbin 则是给root使用的指令。

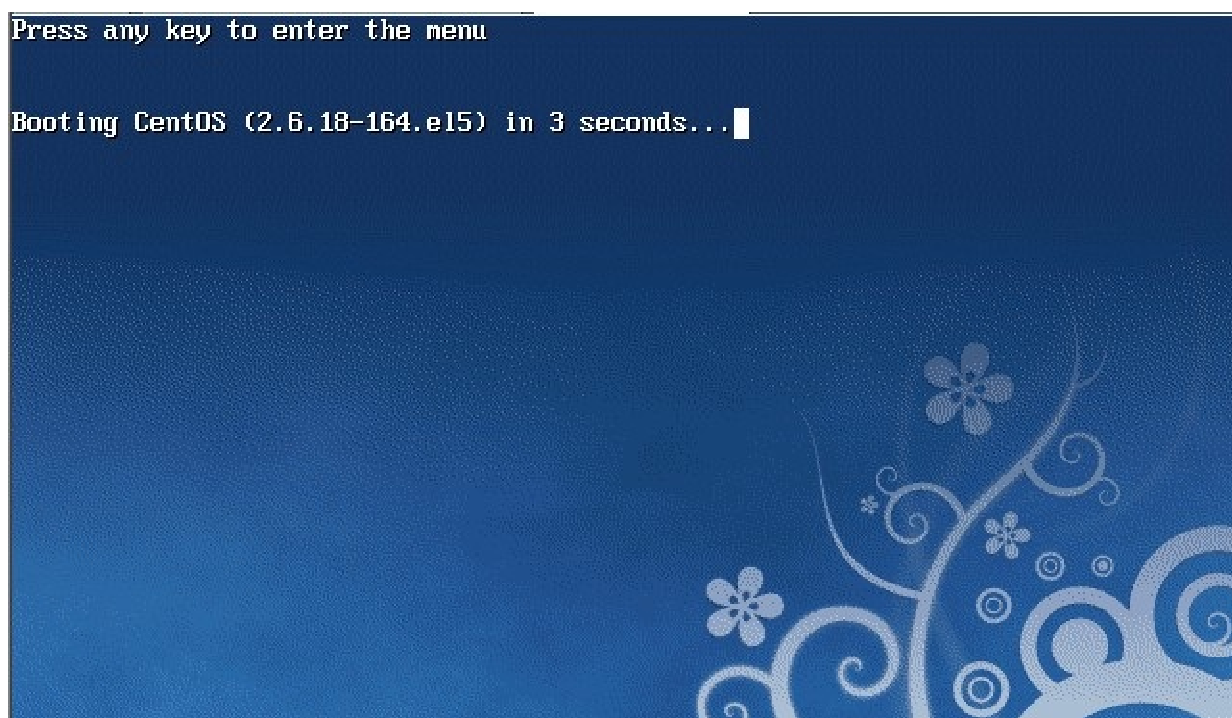
/var： 这是一个非常重要的目录，系统上跑了很多程序，那么每个程序都会有相应的日志产生，而这些日志就被记录到这个目录下，具体在/var/log 目录下，另外mail的预设放置也是在这里。

Linux 忘记密码解决方法

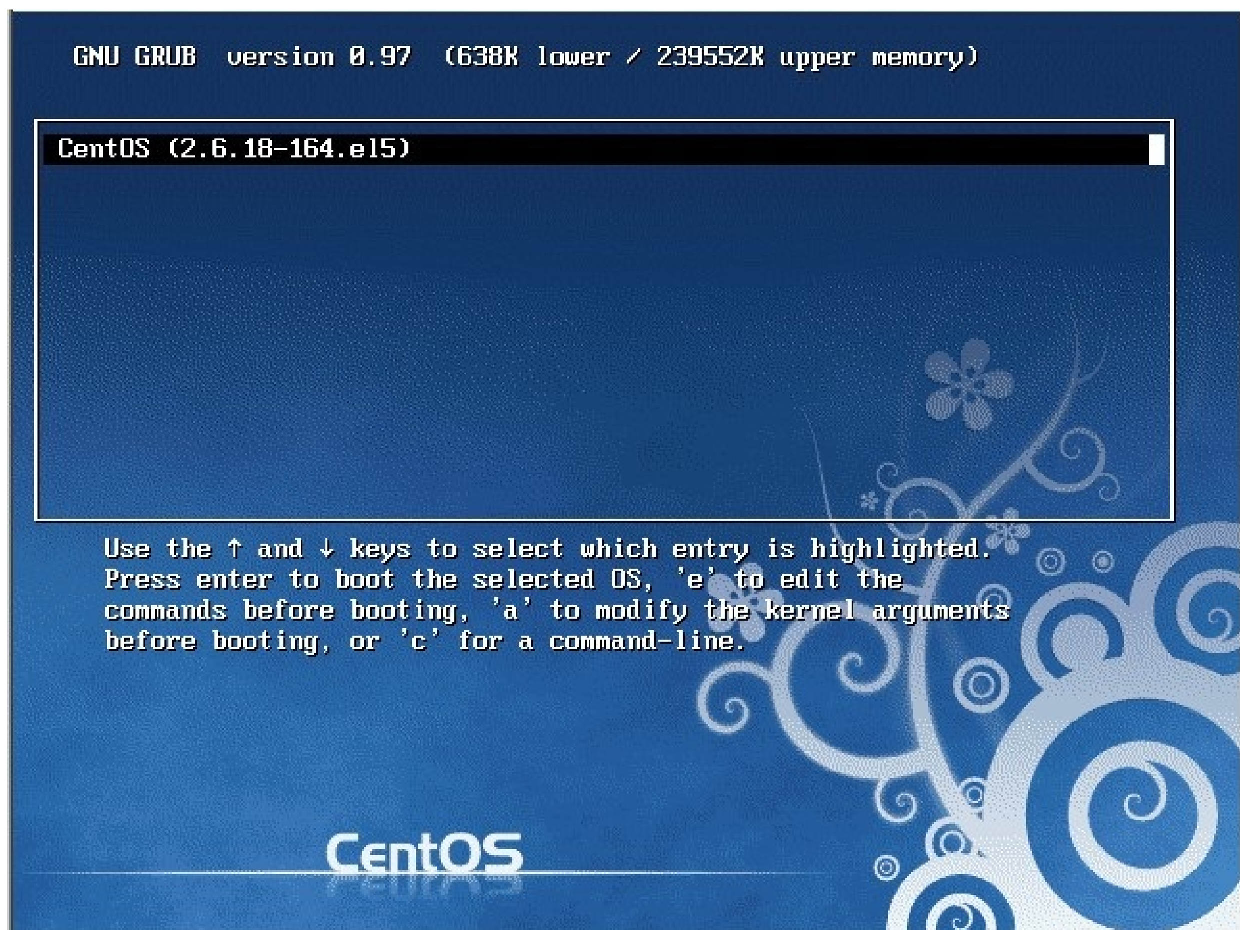
很多朋友经常会忘记Linux系统的root密码，linux系统忘记root密码的情况该怎么办呢？重新安装系统吗？当然不用！进入单用户模式更改一下root密码即可。

步骤如下：

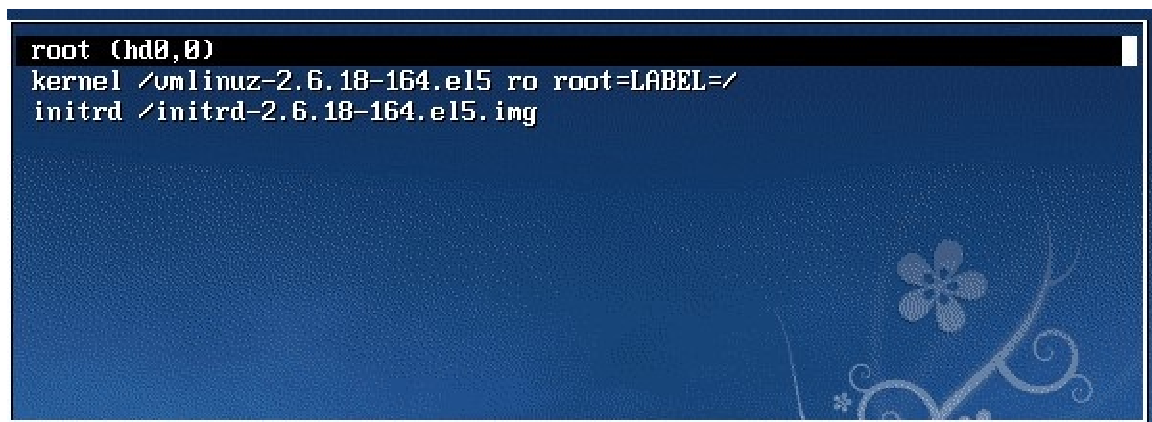
重启linux系统



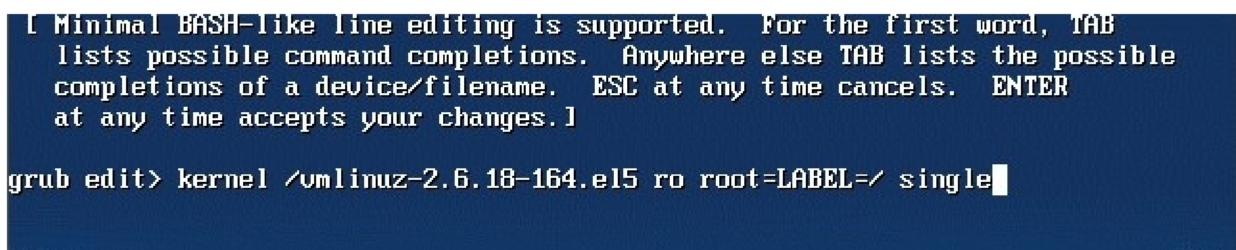
3 秒之内要按一下回车，出现如下界面



然后输入e



在 第二行最后边输入 single，有一个空格。具体方法为按向下尖头移动到第二行，按"e"进入编辑模式



在后边加上single 回车

```
root (hd0,0)
kernel /vmlinuz-2.6.18-164.el5 ro root=LABEL=/ single
initrd /initrd-2.6.18-164.el5.img
```

最后按"b"启动，启动后就进入了单用户模式了

```
no fstab.sys, mounting internal defaults
Switching to new root and running init.
unmounting old /dev
unmounting old /proc
unmounting old /sys
type=1404 audit(1303914022.636:2): enforcing=1 old_enforcing=0 auid=4294967295 s
es=4294967295
type=1403 audit(1303914023.222:3): policy loaded auid=4294967295 ses=4294967295
INIT: version 2.86 booting
                Welcome to  CentOS release 5.4 (Final)
                Press 'I' to enter interactive startup.
Setting clock  (utc): Wed Apr 27 22:20:50 CST 2011        [  OK  ]
Starting udev:                                             [  OK  ]
Loading default keymap (us):                               [  OK  ]
Setting hostname localhost.localdomain:                    [  OK  ]
No devices found
Setting up Logical Volume Management:                      [  OK  ]
Checking filesystems
/: clean, 118508/1969568 files, 713597/1967962 blocks
/boot: clean, 35/26104 files, 14714/104388 blocks
                                                           [  OK  ]
Remounting root filesystem in read-write mode:             [  OK  ]
Mounting local filesystems:                                [  OK  ]
Enabling /etc/fstab swaps:                                 [  OK  ]
sh-3.2# _
```

此时已经进入到单用户模式了，你可以更改root密码了。更密码的命令为 passwd

```
sh-3.2# passwd
Changing password for user root.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

【使用系统安装光盘的救援模式】

救援模式即rescue ，这个模式主要是应用于，系统无法进入的情况。如，grub损坏或者某一个配置文件修改出错。如何使用rescue模式呢？

光盘启动，按F5 进入rescue模式

```
 -  To install or upgrade in graphical mode, press the <ENTER> key.

 -  To install or upgrade in text mode, type: linux text <ENTER>.

 -  Use the function keys listed below for more information.

[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]
boot: _
```

输入linux rescue 回车

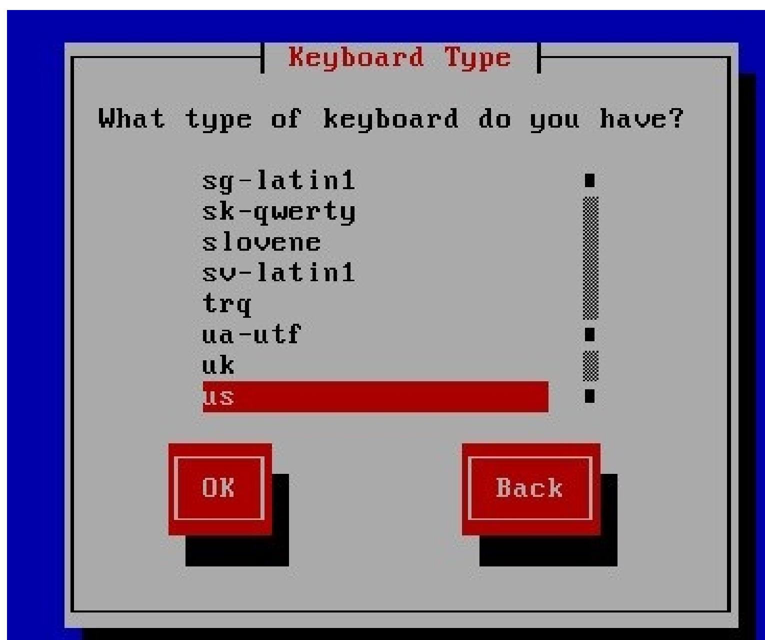
```
[F1-Main] [F2-Options] [F3-General] [F4-Kernel] [F5-Rescue]

boot: linux rescue_
```

选择语言，笔者建议你选择英语

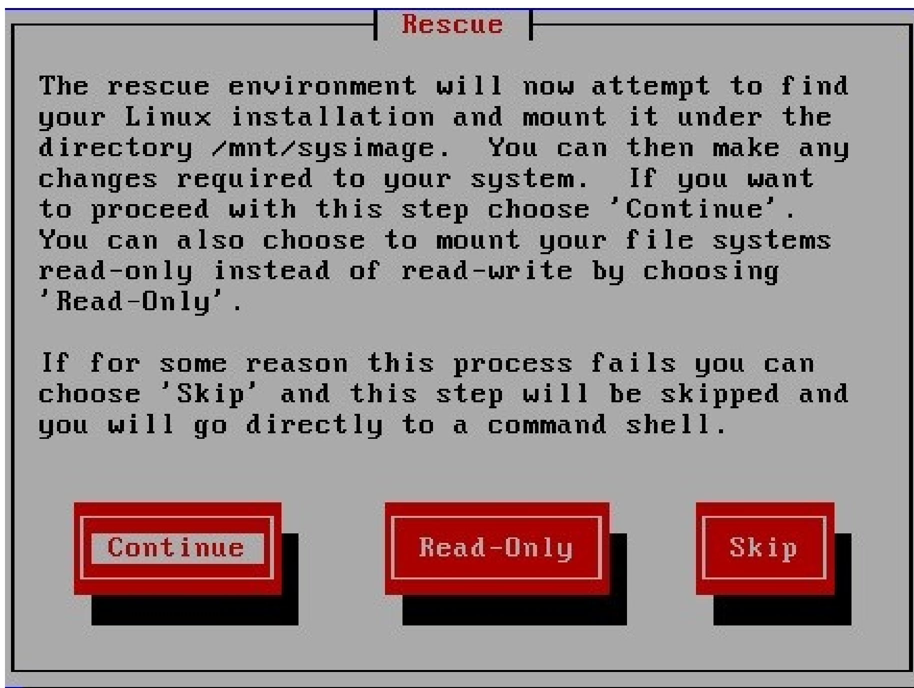


选择us 键盘





这里问你是否启动网络，有时候可能会联网调试。我们选no

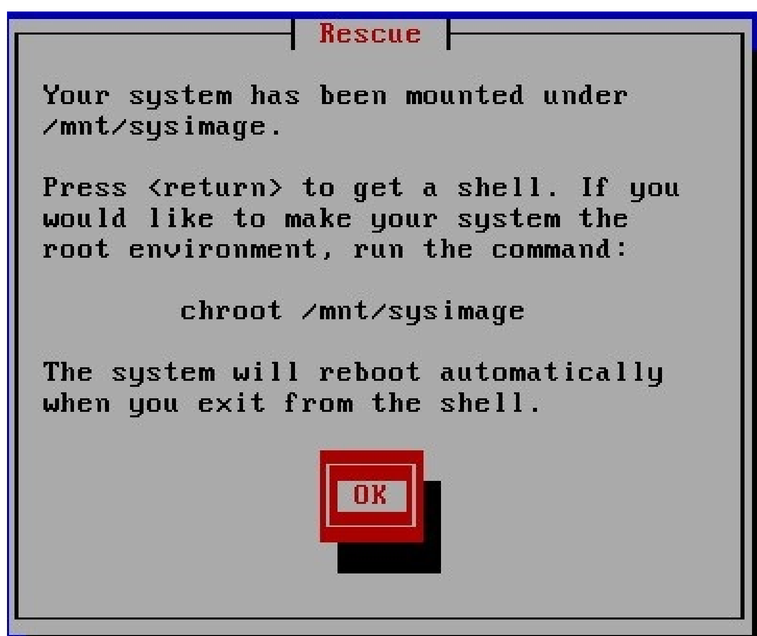


这里告诉我们，接下来会把系统挂载在/mnt/sysimage 中。

其中有三个选项:

- Continue 就是挂载后继续下一步。
- Read-Only 挂载成只读，这样更安全，有时文件系统损坏时，只读模式会防止文件系统近一步损坏。
- Skip就是不挂载，进入一个命令窗口模式。

这里我们选择Continue。



至此，系统已经挂载到了/mnt/sysimage中。接下来回车，输入chroot /mnt/sysimage 进入管理员环境。

```
Your system is mounted under the /mnt/sysimage directory.
When finished please exit from the shell and your system will reboot.

sh-3.2# chroot /mnt/sysimage/
sh-3.2# _
```

提示： 其实也可以到rescue模式下更改root的密码的。这个rescue模式和windows PE系统很相近。

当运行了chroot /mnt/sysimage/ 后，再ls 看到目录结构和原来系统中的目录结构是一样的。

没错！现在的环境和原来系统的环境是一模一样的。你可以输入exit 或者按Ctrl + D退出这个环境。然后你再ls 看一下

```
sh-3.2# ls
bin  etc   lib  modules  proc  sbin     sys  usr
dev  init  mnt  oldtmp   root  selinux  tmp  var
sh-3.2# ls /mnt/
runtime  source  sysimage
sh-3.2# _
```

这个目录其实就是rescue模式下的目录结构，而我们的系统文件全部在 /mnt/sysimage目录下。

Linux 远程登录

Linux一般作为服务器使用，而服务器一般放在机房，你不可能在机房操作你的Linux服务器。

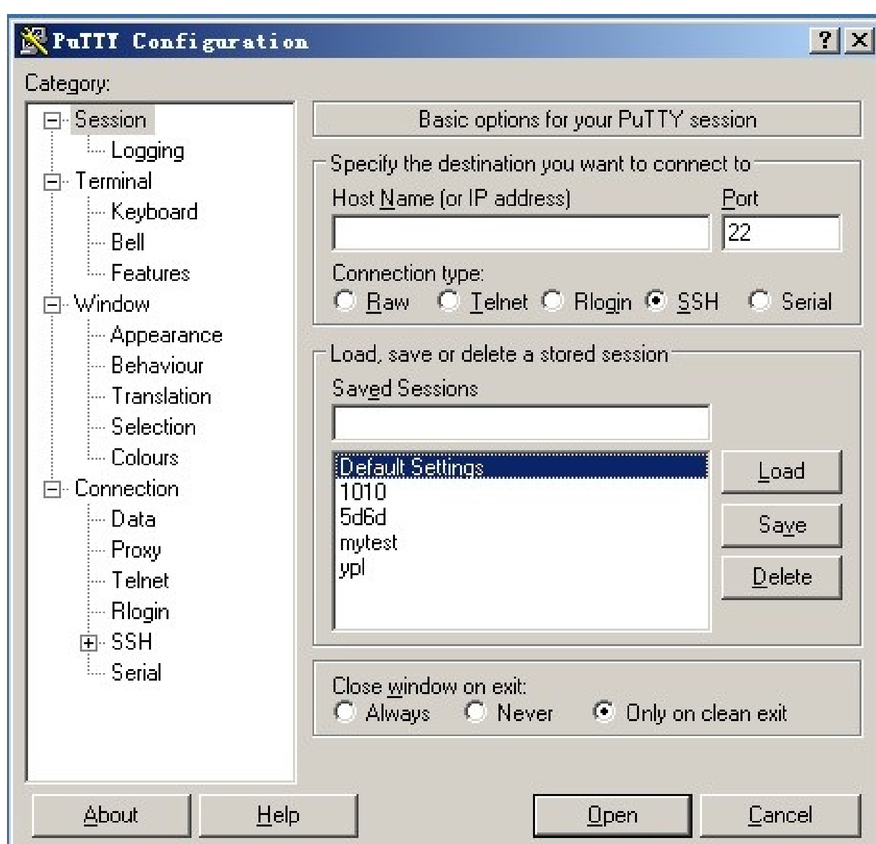
这事我们就需要远程登录到Linux服务器来管理维护系统。

Linux系统中是通过ssh服务实现的远程登录功能，默认ssh服务端口号为 22。

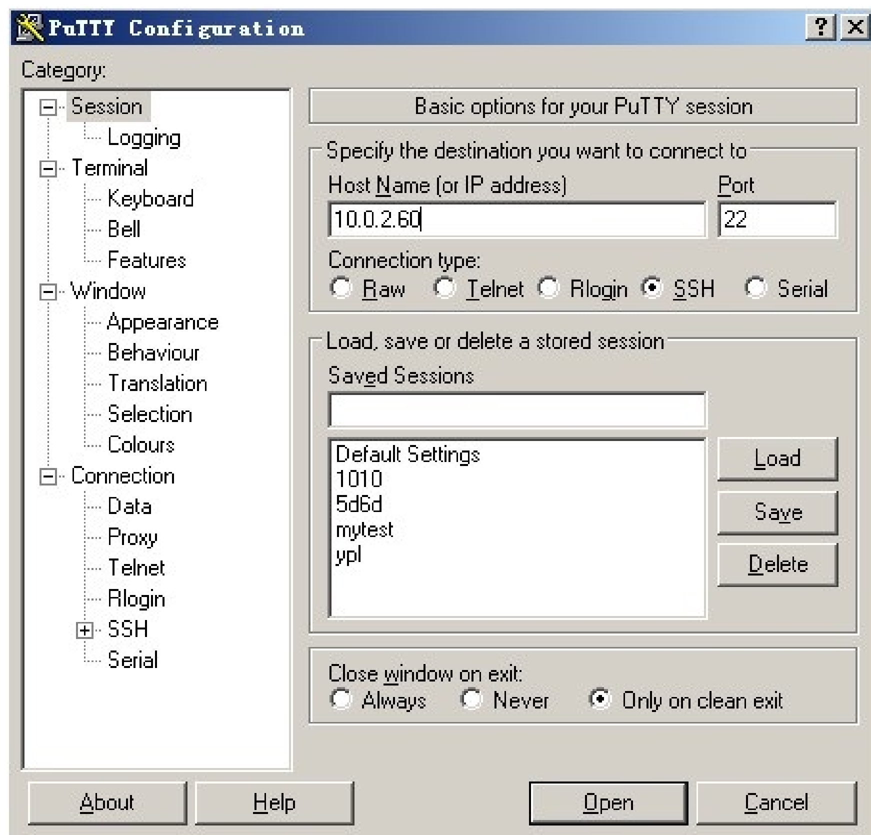
Window系统上 Linux 远程登录客户端有SecureCRT, Putty, SSH Secure Shell等，本文以Putty为例来登录远程服务器。

putty下载地址：http://www.putty.org/

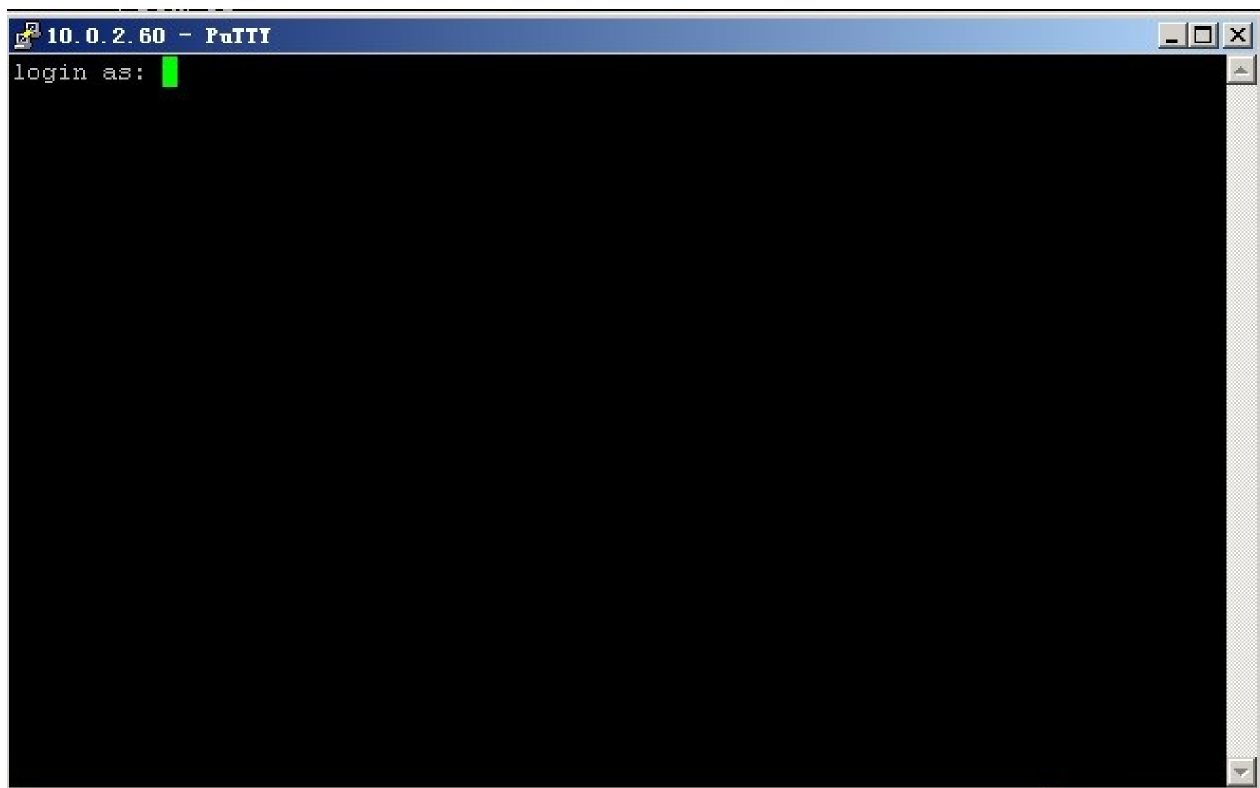
如果你下载了putty，请双击putty.exe 然后弹出如下的窗口。



在Host Name(or IP address) 下面的框中输入你要登录的远程服务器IP(可以通过ifconfig命令查看服务器ip)，然后回车。



此时，提示我们输入要登录的用户名。



输入root 然后回车，再输入密码，就能登录到远程的linux系统了。

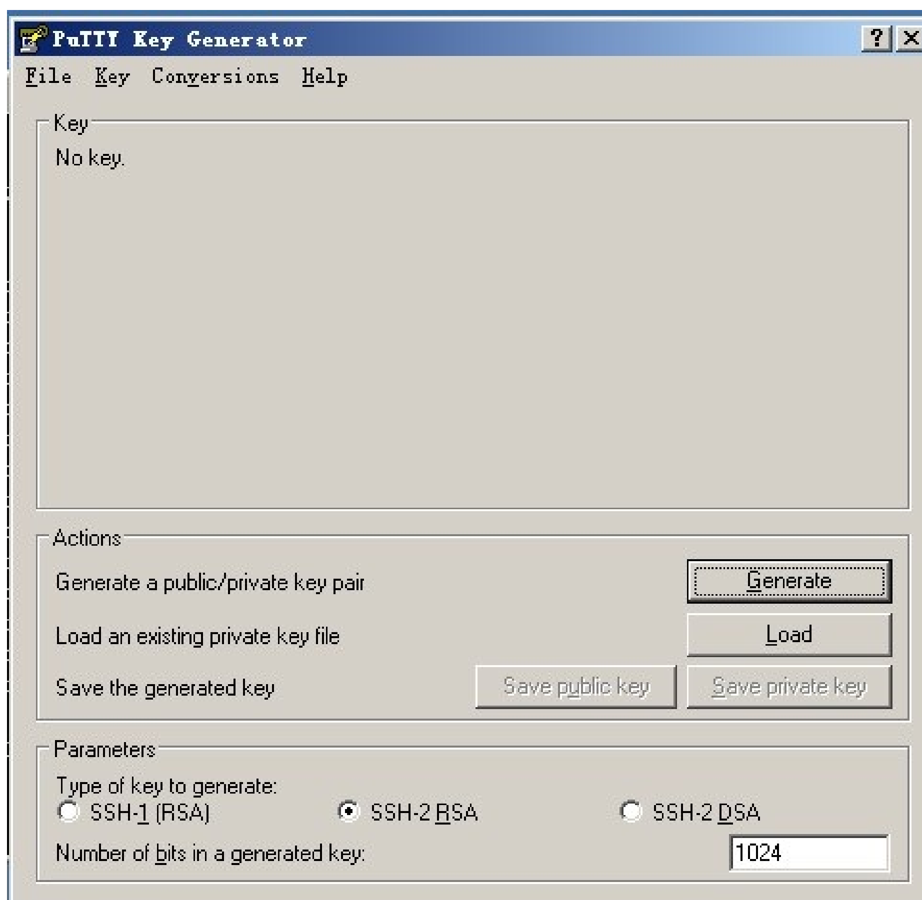
```
root@localhost:~
login as: root
root@10.0.2.60's password:
Last login: Fri Apr 29 14:10:52 2011
[root@localhost ~]#
```

使用密钥认证机制远程登录linux

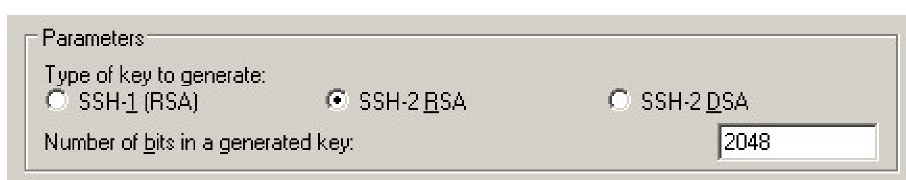
SSH 为 Secure Shell 的缩写，由 IETF 的网络工作小组（Network Working Group）所制定。

SSH 为建立在应用层和传输层基础上的安全协议。

首先使用工具 PUTTYGEN.EXE 生成密钥对。打开工具PUTTYGEN.EXE后如下图所示：



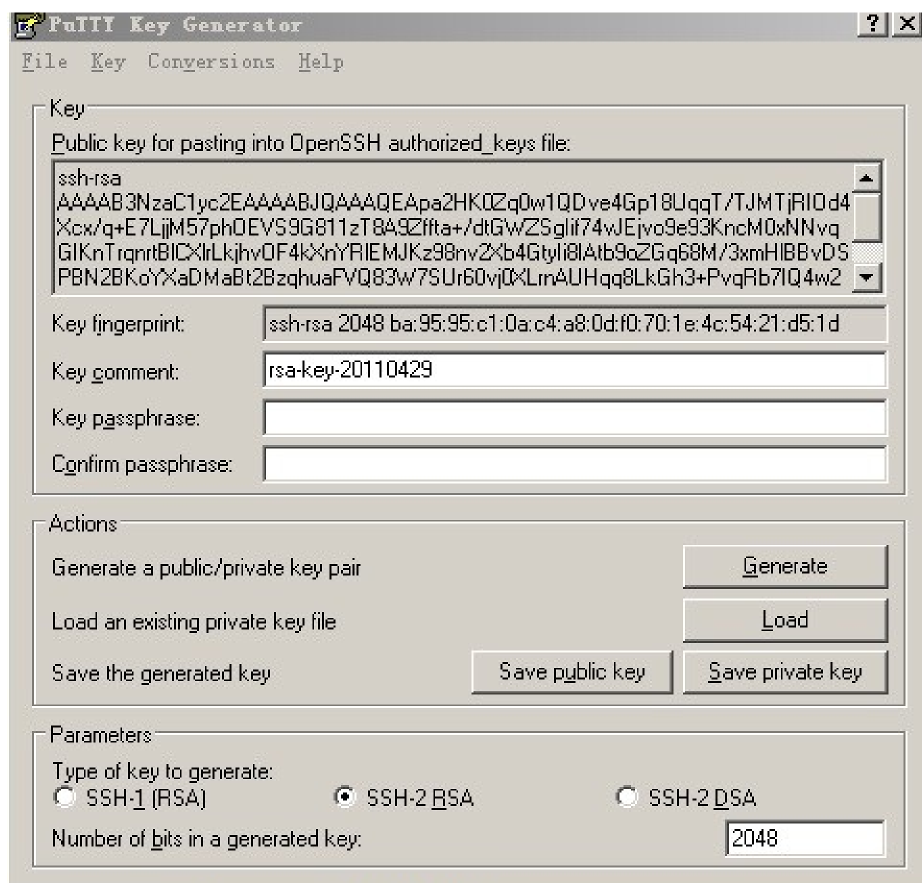
该工具可以生成三种格式的key ：SSH-1(RSA) SSH-2(RSA) SSH-2(DSA)， 我们采用默认的格式即SSH-2(RSA)。Number of bits in a generated key 这个是指生成的key的大小， 这个数值越大， 生成的key就越复杂， 安全性就越高。这里我们写2048.



然后单击Generate 开始生成密钥对：



注意的是，在这个过程中鼠标要来回的动，否则这个进度条是不会动的。



到这里，密钥对已经生成了。你可以给你的密钥输入一个密码，（在Key Passphrase那里）也可以留空。然后点 Save public key 保存公钥，点 Save private Key 保存私钥。笔者建议你放到一个比较安全的地方，一来防止别人偷窥，二来防止误删除。接下来就该到远程linux主机上设置了。

1）创建目录 /root/.ssh 并设置权限

[root@localhost ~]# mkdir /root/.ssh mkdir 命令用来创建目录，以后会详细介绍，暂时只了解即可。

[root@localhost ~]# chmod 700 /root/.ssh chmod 命令是用来修改文件属性权限的，以后会详细介绍。

2）创建文件 / root/.ssh/authorized_keys

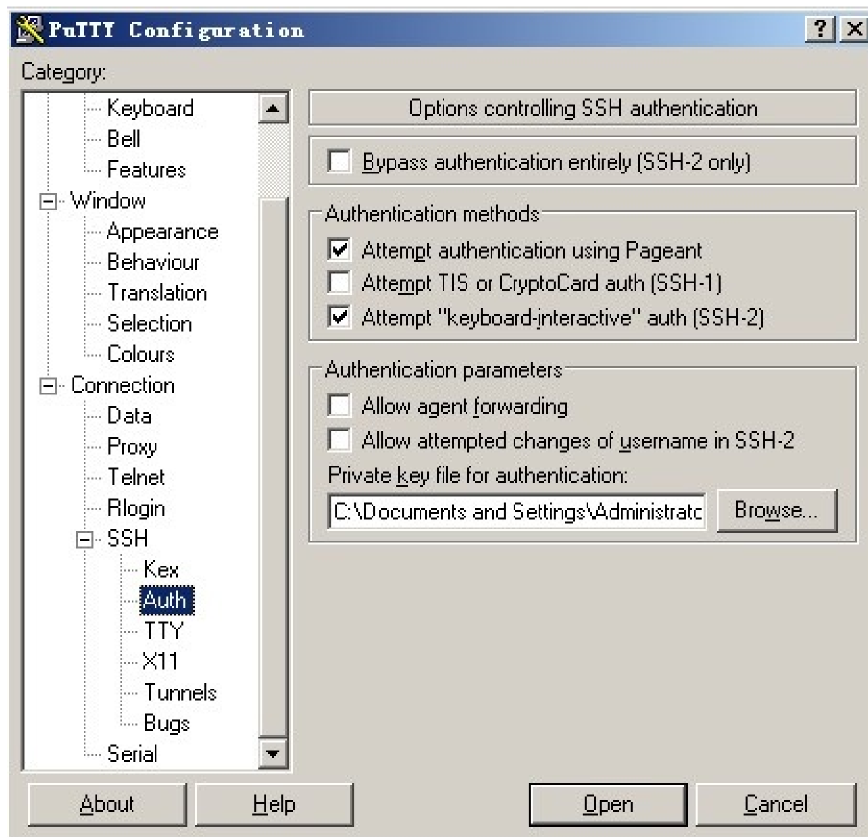
[root@localhost ~]# vim /root/.ssh/authorized_keys vim 命令是编辑一个文本文件的命令，同样在后续章节详细介绍。

3）打开刚才生成的public key 文件，建议使用写字板打开，这样看着舒服一些，复制从AAAA开头至 "---- END SSH2 PUBLIC KEY ----" 该行上的所有内容，粘贴到/root/.ssh/authorized_keys 文件中，要保证所有字符在一行。（可以先把复制的内容拷贝至记事本，然后编辑成一行载粘贴到该文件中）。

在这里要简单介绍一下，如何粘贴，用vim打开那个文件后，该文件不存在，所以vim会自动创建。按一下字母"i"然后同时按shift + Insert 进行粘贴（或者单击鼠标邮件即可），前提是已经复制到剪切板中了。粘贴好后，然后把光标移动到该行最前面输入ssh-ras，然后按空格。再按ESC，然后输入冒号wq 即 :wq 就保存了。格式如下图：

```
root@localhost:~
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEApa2HK0Zq0w1QDve4Gp18UqqT/TJMTjRIOd4Xcx/q+E7LjjM57phOE
VS9G811zT8A9Zffta+/dtGWZSgIif74vJEjvo9e93KncM0xNNvqGIKnTrqnrtBlCXlrLkjhvOF4kXnYRlEMJKz98n
v2Xb4GtyIi8IAtb9oZGq68M/3xmHlBBvDSPBN2BKoYXaDMaBt2BzqhuaFVQ83W7SUr60vj0XLrnAUHqq8LkGh3+Pv
qRb7IQ4w2E9a147WXlIxbRzLQFW7HroAJ3fB622A4H7VvPD95bZZ913xnb+HjBAuxti8uf3/ezm8Via7vzT90VPv2
pQqMvDlnkgDV65eADWu02w==
~
:wq
```

4）再设置putty选项，点窗口左侧的SSh –> Auth，单击窗口右侧的Browse… 选择刚刚生成的私钥，再点Open，此时输入root，就不用输入密码就能登录了。



如果在前面你设置了Key Passphrase， 那么此时就会提示你输入密码的。为了更加安全建议大家要设置一个Key Passphrase。

Linux 文件基本属性

Linux系统是一种典型的多用户系统，不同的用户处于不同的地位，拥有不同的权限。为了保护系统的安全性，Linux系统对不同的用户访问同一文件（包括目录文件）的权限做了不同的规定。

在Linux中我们可以使用ll或者ls –l命令来显示一个文件的属性以及文件所属的用户和组，如：

```
[root@www /]# ls -l
total 64
dr-xr-xr-x   2 root root 4096 Dec 14  2012 bin
dr-xr-xr-x   4 root root 4096 Apr 19  2012 boot
……
```

实例中，bin文件的第一个属性用"d"表示。"d"在Linux中代表该文件是一个目录文件。

在Linux中第一个字符代表这个文件是目录、文件或链接文件等等。

- 当为[*d*]则是目录
- 当为[-]则是文件；
- 若是[*l*]则表示为链接文档(link file)；
- 若是[*b*]则表示为装置文件里面的可供储存的接口设备(可随机存取装置)；
- 若是[*c*]则表示为装置文件里面的串行端口设备，例如键盘、鼠标(一次性读取装置)。

接下来的字符中，以三个为一组，且均为『rwx』 的三个参数的组合。其中，[r]代表可读(read)、[w]代表可写(write)、[x]代表可执行(execute)。 要注意的是，这三个权限的位置不会改变，如果没有权限，就会出现减号[-]而已。

每个文件的属性由左边第一部分的10个字符来确定（如下图）。

文件类型	属主权限	属组权限	其他用户权限
0	1 2 3	4 5 6	7 8 9
d	rwx	r-x	r-x
目录文件	读 写 执行	读 写 执行	读 写 执行

从左至右用0-9这些数字来表示。

第0位确定文件类型，第1-3位确定属主（该文件的所有者）拥有该文件的权限。

第4-6位确定属组（所有者的同组用户）拥有该文件的权限，第7-9位确定其他用户拥有该文件的权限。

其中，第1、4、7位表示读权限，如果用"r"字符表示，则有读权限，如果用"-"字符表示，则没有读权限；

第2、5、8位表示写权限，如果用"w"字符表示，则有写权限，如果用"-"字符表示没有写权限；第3、6、9位表示可执行权限，如果用"x"字符表示，则有执行权限，如果用"-"字符表示，则没有执行权限。

Linux文件属主和属组

```
[root@www /]# ls -l
total 64
dr-xr-xr-x   2 root root 4096 Dec 14  2012 bin
dr-xr-xr-x   4 root root 4096 Apr 19  2012 boot
……
```

对于文件来说，它都有一个特定的所有者，也就是对该文件具有所有权的用户。

同时，在Linux系统中，用户是按组分类的，一个用户属于一个或多个组。

文件所有者以外的用户又可以分为文件所有者的同组用户和其他用户。

因此，Linux系统按文件所有者、文件所有者同组用户和其他用户来规定了不同的文件访问权限。

在以上实例中，bin文件是一个目录文件，属主和属组都为root，属主有可读、可写、可执行的权限；与属主同组的其他用户有可读和可执行的权限；其他用户也有可读和可执行的权限。

更改文件属性

1、chgrp：更改文件属组

语法：

```
chgrp [-R] 属组名文件名
```

参数选项

- -R：递归更改文件属组，就是在更改某个目录文件的属组时，如果加上-R的参数，那么该目录下的所有文件的属组都会更改。

2、chown：更改文件属主，也可以同时更改文件属组

语法：

```
chown [-R] 属主名 文件名
chown [-R] 属主名：属组名 文件名
```

进入 /root 目录（~）将install.log的拥有者改为bin这个账号：

```
[root@www ~] cd ~
[root@www ~]# chown bin install.log
[root@www ~]# ls -l
-rw-r--r--  1 bin  users 68495 Jun 25 08:53 install.log
```

将install.log的拥有者与群组改回为root：

```
[root@www ~]# chown root:root install.log
[root@www ~]# ls -l
-rw-r--r--  1 root root 68495 Jun 25 08:53 install.log
```

3、 chmod：更改文件9个属性

Linux文件属性有两种设置方法，一种是数字，一种是符号。

Linux文件的基本权限就有九个，分别是owner/group/others三种身份各有自己的read/write/execute权限。

先复习一下刚刚上面提到的数据：文件的权限字符为：『-rwxrwxrwx』， 这九个权限是三个三个一组的！其中，我们可以使用数字来代表各个权限，各权限的分数对照表如下：

- r:4
- w:2
- x:1

每种身份(owner/group/others)各自的三个权限(r/w/x)分数是需要累加的，例如当权限为： [-rwxrwx---] 分数则是：

- owner = rwx = 4+2+1 = 7
- group = rwx = 4+2+1 = 7
- others= --- = 0+0+0 = 0

所以等一下我们设定权限的变更时，该文件的权限数字就是770啦！变更权限的指令chmod的语法是这样的：

```
chmod [-R] xyz 文件或目录
```

选项与参数：

- xyz : 就是刚刚提到的数字类型的权限属性，为 rwx 属性数值的相加。
- -R : 进行递归(recursive)的持续变更，亦即连同次目录下的所有文件都会变更

举例来说，如果要将.bashrc这个文件所有的权限都设定启用，那么命令如下：

```
[root@www ~]# ls -al .bashrc
-rw-r--r--  1 root root 395 Jul  4 11:45 .bashrc
[root@www ~]# chmod 777 .bashrc
[root@www ~]# ls -al .bashrc
-rwxrwxrwx  1 root root 395 Jul  4 11:45 .bashrc
```

那如果要将权限变成 *-rwxr-xr--* 呢？那么权限的分数就成为 [4+2+1][4+0+1][4+0+0]=754。

符号类型改变文件权限

还有一个改变权限的方法呦！从之前的介绍中我们可以发现，基本上就九个权限分别是(1)user (2)group (3)others三种身份啦！ 那么我们就可以藉由u, g, o来代表三种身份的权限！

此外， a 则代表 all 亦即全部的身份！那么读写的权限就可以写成r, w, x！也就是可以使用底下的方式来看：

chmod	u g o a	+(加入) -(除去) =(设定)	r w x	文件或目录

如果我们需要将文件权限设置为 -rwxr-xr-- ， 可以使用 *chmod u=rwx,g=rx,o=r* 文件名 来设定:

```
[root@www ~]# ls -al .bashrc
-rwxr-xr-x  1 root root 395 Jul  4 11:45 .bashrc
[root@www ~]# chmod  a+w  .bashrc
[root@www ~]# ls -al .bashrc
-rwxrwxrwx  1 root root 395 Jul  4 11:45 .bashrc
```

而如果是要将权限去掉而不改变其他已存在的权限呢？例如要拿掉全部人的可执行权限，则：

```
[root@www ~]# chmod  a-x  .bashrc
[root@www ~]# ls -al .bashrc
-rw-rw-rw-  1 root root 395 Jul  4 11:45 .bashrc
```

Linux 文件与目录管理

我们知道Linux的目录结构为树状结构，最顶级的目录为根目录 /。

其他目录通过挂载可以将它们添加到树中，通过解除挂载可以移除它们。

在开始本教程前我们需要先知道什么是绝对路径与相对路径。

- 绝对路径：
  路径的写法，由根目录 / 写起，例如： /usr/share/doc 这个目录。
- 相对路径：
  路径的写法，不是由 / 写起，例如由 /usr/share/doc 要到 /usr/share/man 底下时，可以写成： cd ../man 这就是相对路径的写法啦！

处理目录的常用命令

接下来我们就来看几个常见的处理目录的命令吧：

- ls: 列出目录
- cd：切换目录
- pwd：显示目前的目录
- mkdir：创建一个新的目录
- rmdir：删除一个空的目录
- cp: 复制文件或目录
- rm: 移除文件或目录

你可以使用 *man [命令]* 来查看各个命令的使用文档，如 ：man cp。

ls (列出目录)

在Linux系统当中， ls 命令可能是最常被运行的。

语法：

```
[root@www ~]# ls [-aAdfFhilnrRSt] 目录名称
[root@www ~]# ls [--color={never,auto,always}] 目录名称
[root@www ~]# ls [--full-time] 目录名称
```

选项与参数：

- -a ：全部的文件，连同隐藏档(开头为 . 的文件) 一起列出来(常用)
- -d ：仅列出目录本身，而不是列出目录内的文件数据(常用)

- -l ：长数据串列出，包含文件的属性与权限等等数据；(常用)

将家目录下的所有文件列出来(含属性与隐藏档)

```
[root@www ~]# ls -al ~
```

cd (切换目录)

cd是Change Directory的缩写，这是用来变换工作目录的命令。

语法：

```
 cd [相对路径或绝对路径]
```

```
#使用 mkdir 命令创建w3cschool.cc目录
[root@www ~]# mkdir w3cschool.cc

#使用绝对路径切换到w3cschool.cc目录
[root@www ~]# cd /root/w3cschool.cc/

#使用相对路径切换到w3cschool.cc目录
[root@www ~]# cd ./w3cschool.cc/

# 表示回到自己的家目录，亦即是 /root 这个目录
[root@www w3cschool.cc]# cd ~

# 表示去到目前的上一级目录，亦即是 /root 的上一级目录的意思；
[root@www ~]# cd ..
```

接下来大家多操作几次应该就可以很好的理解 cd 命令的。

pwd (显示目前所在的目录)

pwd是Print Working Directory的缩写，也就是显示目前所在目录的命令。

```
[root@www ~]# pwd [-P]
选项与参数：
-P  ：显示出确实的路径，而非使用连结 (link) 路径。

范例：单纯显示出目前的工作目录：
[root@www ~]# pwd
/root   <== 显示出目录啦～

范例：显示出实际的工作目录，而非连结档本身的目录名而已
[root@www ~]# cd /var/mail   <==注意，/var/mail是一个连结档
[root@www mail]# pwd
/var/mail         <==列出目前的工作目录
[root@www mail]# pwd -P
/var/spool/mail   <==怎么回事？有没有加 -P 差很多～
[root@www mail]# ls -ld /var/mail
lrwxrwxrwx 1 root root 10 Sep  4 17:54 /var/mail -> spool/mail
# 看到这里应该知道为啥了吧？因为 /var/mail 是连结档，连结到 /var/spool/mail
# 所以，加上 pwd -P 的选项后，会不以连结档的数据显示，而是显示正确的完整路径啊！
```

mkdir (创建新目录)

如果想要创建新的目录的话，那么就使用mkdir (make directory)吧。

语法：

```
mkdir [-mp] 目录名称
```

选项与参数：

- -m ：配置文件的权限喔！直接配置，不需要看默认权限 (umask) 的脸色～
- -p ：帮助你直接将所需要的目录(包含上一级目录)递回创建起来！

范例：请到/tmp底下尝试创建数个新目录看看：

```
[root@www ~]# cd /tmp
[root@www tmp]# mkdir test    <==创建一名为 test 的新目录
[root@www tmp]# mkdir test1/test2/test3/test4
mkdir: cannot create directory `test1/test2/test3/test4':
No such file or directory       <== 没办法直接创建此目录啊！
[root@www tmp]# mkdir -p test1/test2/test3/test4
```

加了这个 -p 的选项，可以自行帮你创建多层目录！

范例：创建权限为rwx--x--x的目录

```
[root@www tmp]# mkdir -m 711 test2
[root@www tmp]# ls -l
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
```

上面的权限部分，如果没有加上 -m 来强制配置属性，系统会使用默认属性。

如果我们使用 -m ， 如上例我们给予 -m 711 来给予新的目录 drwx--x--x 的权限。

rmdir (删除空的目录)

语法：

```
 rmdir [-p] 目录名称
```

选项与参数：

- **-p** ：连同上一级『空的』目录也一起删除

删除 w3cschool.cc 目录

```
[root@www tmp]# rmdir w3cschool.cc/
```

范例：将於mkdir范例中创建的目录(/tmp底下)删除掉！

```
[root@www tmp]# ls -l   <==看看有多少目录存在？
drwxr-xr-x  3 root  root 4096 Jul 18 12:50 test
drwxr-xr-x  3 root  root 4096 Jul 18 12:53 test1
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
[root@www tmp]# rmdir test   <==可直接删除掉，没问题
[root@www tmp]# rmdir test1  <==因为尚有内容，所以无法删除！
rmdir: `test1': Directory not empty
[root@www tmp]# rmdir -p test1/test2/test3/test4
[root@www tmp]# ls -l        <==您看看，底下的输出中test与test1不见了！
drwx--x--x  2 root  root 4096 Jul 18 12:54 test2
```

利用 -p 这个选项，立刻就可以将 test1/test2/test3/test4 一次删除。

不过要注意的是，这个 rmdir 仅能删除空的目录，你可以使用 rm 命令来删除非空目录。

cp (复制文件或目录)

cp 即拷贝文件和目录。

语法:

```
[root@www ~]# cp [-adfilprsu] 来源档(source) 目标档(destination)
[root@www ~]# cp [options] source1 source2 source3 .... directory
```

选项与参数：

- -a ：相当於 -pdr 的意思，至於 pdr 请参考下列说明；(常用) *-d* ：若来源档为连结档的属性*(link file)*，则复制连结档属性而非文件本身； -f ：为强制(force)的意思，若目标文件已经存在且无法开启，则移除后再尝试一次； *-i* ：若目标档*(destination)*已经存在时，在覆盖时会先询问动作的进行*(*常用*)* -l ：进行硬式连结(hard link)的连结档创建，而非复制文件本身； *-p* ：连同文件的属性一起复制过去，而非使用默认属性*(*备份常用*)*； -r ：递回持续复制，用於目录的复制行为；(常用) *-s* ：复制成为符号连结档 *(symbolic link)*，亦即『捷径』文件； -u ：若 destination 比 source 旧才升级 destination ！

用root身份，将家目录下的 .bashrc 复制到 /tmp 下，并更名为 bashr

```
[root@www ~]# cp ~/.bashrc /tmp/bashrc
[root@www ~]# cp -i ~/.bashrc /tmp/bashrc
cp: overwrite `/tmp/bashrc'? n  <==n不覆盖，y为覆盖
```

rm (移除文件或目录)

语法：

```
rm [-fir] 文件或目录
```

选项与参数：

- -f ：就是 force 的意思，忽略不存在的文件，不会出现警告信息；
- -i ：互动模式，在删除前会询问使用者是否动作
- -r ：递回删除啊！最常用在目录的删除了！这是非常危险的选项！！！

将刚刚在 cp 的范例中创建的 bashrc 删除掉！

```
[root@www tmp]# rm -i bashrc
rm: remove regular file `bashrc'? y
```

如果加上 -i 的选项就会主动询问喔，避免你删除到错误的档名！

mv (移动文件与目录，或修改名称)

语法：

```
[root@www ~]# mv [-fiu] source destination
[root@www ~]# mv [options] source1 source2 source3 .... directory
```

选项与参数：

- -f ：force 强制的意思，如果目标文件已经存在，不会询问而直接覆盖；
- -i ：若目标文件 (destination) 已经存在时，就会询问是否覆盖！
- -u ：若目标文件已经存在，且 source 比较新，才会升级 (update)

复制一文件，创建一目录，将文件移动到目录中

```
[root@www ~]# cd /tmp
[root@www tmp]# cp ~/.bashrc bashrc
[root@www tmp]# mkdir mvtest
[root@www tmp]# mv bashrc mvtest
```

将某个文件移动到某个目录去，就是这样做！

将刚刚的目录名称更名为 mvtest2

```
[root@www tmp]# mv mvtest mvtest2
```

Linux 文件内容查看

Linux系统中使用以下命令来查看文件的内容：

- cat 由第一行开始显示文件内容
- tac 从最后一行开始显示，可以看出 tac 是 cat 的倒著写！
- nl 显示的时候，顺道输出行号！
- more 一页一页的显示文件内容
- less 与 more 类似，但是比 more 更好的是，他可以往前翻页！
- head 只看头几行
- tail 只看尾巴几行

你可以使用 *man [命令]*来查看各个命令的使用文档，如 ：man cp。

cat

由第一行开始显示文件内容

语法：

```
cat [-AbEnTv]
```

选项与参数：

- -A ：相当於 -vET 的整合选项，可列出一些特殊字符而不是空白而已；
- -b ：列出行号，仅针对非空白行做行号显示，空白行不标行号！
- -E ：将结尾的断行字节 $ 显示出来；
- -n ：列印出行号，连同空白行也会有行号，与 -b 的选项不同；
- -T ：将 [tab] 按键以 ^I 显示出来；
- -v ：列出一些看不出来的特殊字符

检看 /etc/issue 这个文件的内容：

```
[root@www ~]# cat /etc/issue
CentOS release 6.4 (Final)
Kernel \r on an \m
```

tac

tac与cat命令刚好相反，文件内容从最后一行开始显示，可以看出 tac 是 cat 的倒着写！如：

```
[root@www ~]# tac /etc/issue

Kernel \r on an \m
CentOS release 6.4 (Final)
```

nl

显示行号

语法：

```
nl [-bnw] 文件
```

选项与参数：

- -b ：指定行号指定的方式，主要有两种：
  -b a ：表示不论是否为空行，也同样列出行号(类似 cat -n)；
  -b t ：如果有空行，空的那一行不要列出行号(默认值)；
- -n ：列出行号表示的方法，主要有三种：
  -n ln ：行号在萤幕的最左方显示；
  -n rn ：行号在自己栏位的最右方显示，且不加 0 ；
  -n rz ：行号在自己栏位的最右方显示，且加 0 ；
- -w ：行号栏位的占用的位数。

范例一：用 nl 列出 /etc/issue 的内容

```
[root@www ~]# nl /etc/issue
     1  CentOS release 6.4 (Final)
     2  Kernel \r on an \m
```

more

一页一页翻动

```
[root@www ~]# more /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
--More--(28%)  <== 重点在这一行喔！你的光标也会在这里等待你的命令
```

在 more 这个程序的运行过程中，你有几个按键可以按的：

- 空白键 (space)：代表向下翻一页；
- Enter ：代表向下翻『一行』；
- /字串 ：代表在这个显示的内容当中，向下搜寻『字串』这个关键字；
- :f ：立刻显示出档名以及目前显示的行数；
- q ：代表立刻离开 more ，不再显示该文件内容。
- b 或 [ctrl]-b ：代表往回翻页，不过这动作只对文件有用，对管线无用。

less

一页一页翻动，以下实例输出/etc/man.config文件的内容：

```
[root@www ~]# less /etc/man.config
#
# Generated automatically from man.conf.in by the
# configure script.
#
# man.conf from man-1.6d
....(中间省略)....
:   <== 这里可以等待你输入命令！
```

less运行时可以输入的命令有：

- 空白键 ：向下翻动一页；
- [pagedown]：向下翻动一页；
- [pageup] ：向上翻动一页；
- /字串 ：向下搜寻『字串』的功能；
- ?字串 ：向上搜寻『字串』的功能；
- n ：重复前一个搜寻 (与 / 或 ? 有关！)
- N ：反向的重复前一个搜寻 (与 / 或 ? 有关！)
- q ：离开 less 这个程序；

head

取出文件前面几行

语法：

```
head [-n number] 文件
```

选项与参数：

- -n ：后面接数字，代表显示几行的意思

```
[root@www ~]# head /etc/man.config
```

默认的情况中，显示前面 10 行！若要显示前 20 行，就得要这样：

```
[root@www ~]# head -n 20 /etc/man.config
```

tail

取出文件后面几行

语法：

```
tail [-n number] 文件
```

选项与参数：

- -n ：后面接数字，代表显示几行的意思
- -f ：表示持续侦测后面所接的档名，要等到按下[ctrl]-c才会结束tail的侦测

```
[root@www ~]# tail /etc/man.config
# 默认的情况中，显示最后的十行！若要显示最后的 20 行，就得要这样：
[root@www ~]# tail -n 20 /etc/man.config
```

Linux 用户和用户组管理

Linux系统是一个多用户多任务的分时操作系统，任何一个要使用系统资源的用户，都必须首先向系统管理员申请一个账号，然后以这个账号的身份进入系统。

用户的账号一方面可以帮助系统管理员对使用系统的用户进行跟踪，并控制他们对系统资源的访问；另一方面也可以帮助用户组织文件，并为用户提供安全性保护。

每个用户账号都拥有一个惟一的用户名和各自的口令。

用户在登录时键入正确的用户名和口令后，就能够进入系统和自己的主目录。

实现用户账号的管理，要完成的工作主要有如下几个方面：

- 用户账号的添加、删除与修改。
- 用户口令的管理。
- 用户组的管理。

一、Linux系统用户账号的管理

用户账号的管理工作主要涉及到用户账号的添加、修改和删除。

添加用户账号就是在系统中创建一个新账号，然后为新账号分配用户号、用户组、主目录和登录Shell等资源。刚添加的账号是被锁定的，无法使用。

1、添加新的用户账号使用useradd命令，其语法如下：

```
useradd 选项 用户名
```

参数说明：

- 选项:
    - -c comment 指定一段注释性描述。
    - -d 目录 指定用户主目录，如果此目录不存在，则同时使用-m选项，可以创建主目录。
    - -g 用户组 指定用户所属的用户组。
    - -G 用户组，用户组 指定用户所属的附加组。
    - -s Shell文件 指定用户的登录Shell。
    - -u 用户号 指定用户的用户号，如果同时有-o选项，则可以重复使用其他用户的标识号。
- 用户名:

指定新账号的登录名。

实例1

```
# useradd –d /usr/sam -m sam
```

此命令创建了一个用户sam，其中-d和-m选项用来为登录名sam产生一个主目录/usr/sam（/usr为默认的用户主目录所在的父目录）。

实例2

```
# useradd -s /bin/sh -g group –G adm,root gem
```

此命令新建了一个用户gem，该用户的登录Shell是 `/bin/sh`， 它属于group用户组，同时又属于adm和root用户组，其中group用户组是其主组。

这里可能新建组：`#groupadd group及groupadd adm`

增加用户账号就是在/etc/passwd文件中为新用户增加一条记录，同时更新其他系统文件如/etc/shadow, /etc/group等。

Linux提供了集成的系统管理工具userconf，它可以用来对用户账号进行统一管理。

3、删除帐号

如果一个用户的账号不再使用，可以从系统中删除。删除用户账号就是要将/etc/passwd等系统文件中的该用户记录删除，必要时还删除用户的主目录。

删除一个已有的用户账号使用`userdel`命令，其格式如下：

```
userdel 选项 用户名
```

常用的选项是-r，它的作用是把用户的主目录一起删除。

例如：

```
# userdel sam
```

此命令删除用户sam在系统文件中（主要是/etc/passwd, /etc/shadow, /etc/group等）的记录，同时删除用户的主目录。

4、修改帐号

修改用户账号就是根据实际情况更改用户的有关属性，如用户号、主目录、用户组、登录Shell等。

修改已有用户的信息使用 `usermod` 命令，其格式如下：

```
usermod 选项 用户名
```

常用的选项包括 `-c, -d, -m, -g, -G, -s, -u以及-o等` ，这些选项的意义与 `useradd` 命令中的选项一样，可以为用户指定新的资源值。

另外，有些系统可以使用选项：-l 新用户名

这个选项指定一个新的账号，即将原来的用户名改为新的用户名。

例如：

```
# usermod -s /bin/ksh -d /home/z –g developer sam
```

此命令将用户sam的登录Shell修改为ksh，主目录改为/home/z，用户组改为developer。

5、用户口令的管理

用户管理的一项重要内容是用户口令的管理。用户账号刚创建时没有口令，但是被系统锁定，无法使用，必须为其指定口令后才可以使用，即使是指定空口令。

指定和修改用户口令的Shell命令是 `passwd` 。超级用户可以为自己和其他用户指定口令，普通用户只能用它修改自己的口令。命令的格式为：

```
passwd 选项 用户名
```

可使用的选项：

- -l 锁定口令，即禁用账号。
- -u 口令解锁。
- -d 使账号无口令。
- -f 强迫用户下次登录时修改口令。

如果默认用户名，则修改当前用户的口令。

例如，假设当前用户是sam，则下面的命令修改该用户自己的口令：

```
$ passwd
Old password:******
New password:*******
Re-enter new password:*******
```

如果是超级用户，可以用下列形式指定任何用户的口令：

```
# passwd sam
New password:*******
Re-enter new password:*******
```

普通用户修改自己的口令时，passwd命令会先询问原口令，验证后再要求用户输入两遍新口令，如果两次输入的口令一致，则将这个口令指定给用户；而超级用户为用户指定口令时，就不需要知道原口令。

为了系统安全起见，用户应该选择比较复杂的口令，例如最好使用8位长的口令，口令中包含有大写、小写字母和数字，并且应该与姓名、生日等不相同。

为用户指定空口令时，执行下列形式的命令：

```
# passwd -d sam
```

此命令将用户sam的口令删除，这样用户sam下一次登录时，系统就不再询问口令。

passwd命令还可以用-l(lock)选项锁定某一用户，使其不能登录，例如：

```
# passwd -l sam
```

二、Linux系统用户组的管理

每个用户都有一个用户组，系统可以对一个用户组中的所有用户进行集中管理。不同Linux 系统对用户组的规定有所不同，如Linux下的用户属于与它同名的用户组，这个用户组在创建用户时同时创建。

用户组的管理涉及用户组的添加、删除和修改。组的增加、删除和修改实际上就是对/etc/group文件的更新。

1、增加一个新的用户组使用groupadd命令。其格式如下：

```
groupadd 选项 用户组
```

可以使用的选项有：

- -g GID 指定新用户组的组标识号（GID）。
- -o 一般与-g选项同时使用，表示新用户组的GID可以与系统已有用户组的GID相同。

实例**1**：

```
# groupadd group1
```

此命令向系统中增加了一个新组group1，新组的组标识号是在当前已有的最大组标识号的基础上加1。

实例**2**：

```
# groupadd -g 101 group2
```

此命令向系统中增加了一个新组group2，同时指定新组的组标识号是101。

2、如果要删除一个已有的用户组，使用**groupdel**命令，其格式如下：

```
groupdel 用户组
```

例如：

```
# groupdel group1
```

此命令从系统中删除组group1。

3、修改用户组的属性使用**groupmod**命令。其语法如下：

```
groupmod 选项 用户组
```

常用的选项有：

- -g GID 为用户组指定新的组标识号。
- -o 与-g选项同时使用，用户组的新GID可以与系统已有用户组的GID相同。
- -n新用户组 将用户组的名字改为新名字

实例**1**：

```
# groupmod -g 102 group2
```

此命令将组group2的组标识号修改为102。

实例**2**：

```
# groupmod –g 10000 -n group3 group2
```

此命令将组group2的标识号改为10000，组名修改为group3。

4、如果一个用户同时属于多个用户组，那么用户可以在用户组之间切换，以便具有其他用户组的权限。

用户可以在登录后，使用命令newgrp切换到其他用户组，这个命令的参数就是目的用户组。例如：

```
$ newgrp root
```

这条命令将当前用户切换到root用户组，前提条件是root用户组确实是该用户的主组或附加组。类似于用户账号的管理，用户组的管理也可以通过集成的系统管理工具来完成。

三、与用户账号有关的系统文件

完成用户管理的工作有许多种方法，但是每一种方法实际上都是对有关的系统文件进行修改。

与用户和用户组相关的信息都存放在一些系统文件中，这些文件包括/etc/passwd, /etc/shadow, /etc/group等。

下面分别介绍这些文件的内容。

1、/etc/passwd文件是用户管理工作涉及的最重要的一个文件。

Linux系统中的每个用户都在/etc/passwd文件中有一个对应的记录行，它记录了这个用户的一些基本属性。

这个文件对所有用户都是可读的。它的内容类似下面的例子：

```
# cat /etc/passwd

root:x:0:0:Superuser:/:
daemon:x:1:1:System daemons:/etc:
bin:x:2:2:Owner of system commands:/bin:
sys:x:3:3:Owner of system files:/usr/sys:
adm:x:4:4:System accounting:/usr/adm:
uucp:x:5:5:UUCP administrator:/usr/lib/uucp:
auth:x:7:21:Authentication administrator:/tcb/files/auth:
cron:x:9:16:Cron daemon:/usr/spool/cron:
listen:x:37:4:Network daemon:/usr/net/nls:
lp:x:71:18:Printer administrator:/usr/spool/lp:
sam:x:200:50:Sam san:/usr/sam:/bin/sh
```

从上面的例子我们可以看到，/etc/passwd中一行记录对应着一个用户，每行记录又被冒号(:)分隔为7个字段，其格式和具体含义如下：

```
用户名:口令:用户标识号:组标识号:注释性描述:主目录:登录Shell
```

1）"用户名"是代表用户账号的字符串。

通常长度不超过8个字符，并且由大小写字母和/或数字组成。登录名中不能有冒号(:)，因为冒号在这里是分隔符。

为了兼容起见，登录名中最好不要包含点字符(.)，并且不使用连字符(-)和加号(+)打头。

2）“口令”一些系统中，存放着加密后的用户口令字。

虽然这个字段存放的只是用户口令的加密串，不是明文，但是由于/etc/passwd文件对所有用户都可读，所以这仍是一个安全隐患。因此，现在许多Linux 系统（如SVR4）都使用了shadow技术，把真正的加密后的用户口令字存放到/etc/shadow文件中，而在/etc/passwd文件的口令字段中只存放一个特殊的字符，例如“x”或者“*”。

3）“用户标识号”是一个整数，系统内部用它来标识用户。

一般情况下它与用户名是一一对应的。如果几个用户名对应的用户标识号是一样的，系统内部将把它们视为同一个用户，但是它们可以有不同的口令、不同的主目录以及不同的登录Shell等。

通常用户标识号的取值范围是0～65 535。0是超级用户root的标识号，1～99由系统保留，作为管理账号，普通用户的标识号从100开始。在Linux系统中，这个界限是500。

4）“组标识号”字段记录的是用户所属的用户组。

它对应着/etc/group文件中的一条记录。

5)“注释性描述”字段记录着用户的一些个人情况。

例如用户的真实姓名、电话、地址等，这个字段并没有什么实际的用途。在不同的Linux 系统中，这个字段的格式并没有统一。在许多Linux系统中，这个字段存放的是一段任意的注释性描述文字，用做finger命令的输出。

6)“主目录”，也就是用户的起始工作目录。

它是用户在登录到系统之后所处的目录。在大多数系统中，各用户的主目录都被组织在同一个特定的目录下，而用户主目录的名称就是该用户的登录名。各用户对自己的主目录有读、写、执行（搜索）权限，其他用户对此目录的访问权限则根据具体情况设置。

7)用户登录后，要启动一个进程，负责将用户的操作传给内核，这个进程是用户登录到系统后运行的命令解释器或某个特定的程序，即Shell。

Shell是用户与Linux系统之间的接口。Linux的Shell有许多种，每种都有不同的特点。常用的有sh(Bourne Shell), csh(C Shell), ksh(Korn Shell), tcsh(TENEX/TOPS-20 type C Shell), bash(Bourne Again Shell)等。

系统管理员可以根据系统情况和用户习惯为用户指定某个Shell。如果不指定Shell，那么系统使用sh为默认的登录Shell，即这个字段的值为/bin/sh。

用户的登录Shell也可以指定为某个特定的程序（此程序不是一个命令解释器）。

利用这一特点，我们可以限制用户只能运行指定的应用程序，在该应用程序运行结束后，用户就自动退出了系统。有些Linux 系统要求只有那些在系统中登记了的程序才能出现在这个字段中。

8)系统中有一类用户称为伪用户（psuedo users）。

这些用户在/etc/passwd文件中也占有一条记录，但是不能登录，因为它们的登录Shell为空。它们的存在主要是方便系统管理，满足相应的系统进程对文件属主的要求。

常见的伪用户如下所示：

```
伪 用 户 含 义
bin 拥有可执行的用户命令文件
sys 拥有系统文件
adm 拥有帐户文件
uucp UUCP使用
lp lp或lpd子系统使用
nobody NFS使用
```

拥有帐户文件

1、除了上面列出的伪用户外，还有许多标准的伪用户，例如：audit, cron, mail, usenet等，它们也都各自为相关的进程和文件所需要。

由于/etc/passwd文件是所有用户都可读的，如果用户的密码太简单或规律比较明显的话，一台普通的计算机就能够很容易地将它破解，因此对安全性要求较高的Linux系统都把加密后的口令字分离出来，单独存放在一个文件中，这个文件是/etc/shadow文件。 有超级用户才拥有该文件读权限，这就保证了用户密码的安全性。

2、/etc/shadow中的记录行与/etc/passwd中的一一对应，它由pwconv命令根据/etc/passwd中的数据自动产生

它的文件格式与/etc/passwd类似，由若干个字段组成，字段之间用":"隔开。这些字段是：

```
登录名:加密口令:最后一次修改时间:最小时间间隔:最大时间间隔:警告时间:不活动时间:失效时间:标志
```

1. "登录名"是与/etc/passwd文件中的登录名相一致的用户账号
2. "口令"字段存放的是加密后的用户口令字，长度为13个字符。如果为空，则对应用户没有口令，登录时不需要口令；如果含有不属于集合 { ./0-9A-Za-z }中的字符，则对应的用户不能登录。
3. "最后一次修改时间"表示的是从某个时刻起，到用户最后一次修改口令时的天数。时间起点对不同的系统可能不一样。例如在SCO Linux 中，这个时间起点是1970年1月1日。
4. "最小时间间隔"指的是两次修改口令之间所需的最小天数。
5. "最大时间间隔"指的是口令保持有效的最大天数。
6. "警告时间"字段表示的是从系统开始警告用户到用户密码正式失效之间的天数。
7. "不活动时间"表示的是用户没有登录活动但账号仍能保持有效的最大天数。
8. "失效时间"字段给出的是一个绝对的天数，如果使用了这个字段，那么就给出相应账号的生存期。期满后，该账号就不再是一个合法的账号，也就不能再用来登录了。

下面是/etc/shadow的一个例子：

```
# cat /etc/shadow

root:Dnakfw28zf38w:8764:0:168:7:::
daemon:*::0:0::::
bin:*::0:0::::
sys:*::0:0::::
adm:*::0:0::::
uucp:*::0:0::::
nuucp:*::0:0::::
auth:*::0:0::::
cron:*::0:0::::
listen:*::0:0::::
lp:*::0:0::::
sam:EkdiSECLWPdSa:9740:0:0::::
```

3、用户组的所有信息都存放在/etc/group文件中。

将用户分组是Linux 系统中对用户进行管理及控制访问权限的一种手段。

每个用户都属于某个用户组；一个组中可以有多个用户，一个用户也可以属于不同的组。

当一个用户同时是多个组中的成员时，在/etc/passwd文件中记录的是用户所属的主组，也就是登录时所属的默认组，而其他组称为附加组。

用户要访问属于附加组的文件时，必须首先使用newgrp命令使自己成为所要访问的组中的成员。

用户组的所有信息都存放在/etc/group文件中。此文件的格式也类似于/etc/passwd文件，由冒号(:)隔开若干个字段，这些字段有：

```
组名:口令:组标识号:组内用户列表
```

1. "组名"是用户组的名称，由字母或数字构成。与/etc/passwd中的登录名一样，组名不应重复。
2. "口令"字段存放的是用户组加密后的口令字。一般Linux 系统的用户组都没有口令，即这个字段一般为空，或者是*。
3. "组标识号"与用户标识号类似，也是一个整数，被系统内部用来标识组。
4. "组内用户列表"是属于这个组的所有用户的列表/b]，不同用户之间用逗号(,)分隔。这个用户组可能是用户的主组，也可能是附加组。

/etc/group文件的一个例子如下：

```
root::0:root
bin::2:root,bin
sys::3:root,uucp
adm::4:root,adm
daemon::5:root,daemon
lp::7:root,lp
users::20:root,sam
```

四、添加量用户批

添加和删除用户对每位Linux系统管理员都是轻而易举的事，比较棘手的是如果要添加几十个、上百个甚至上千个用户时，我们不太可能还使用useradd一个一个地添加，必然要找一种简便的创建大量用户的方法。Linux系统提供了创建大量用户的工具，可以让您立即创建大量用户，方法如下：

（1）先编辑一个文本用户文件。

每一列按照 `/etc/passwd` 密码文件的格式书写，要注意每个用户的用户名、UID、宿主目录都不可以相同，其中密码栏可以留做空白或输入x号。一个范例文件user.txt内容如下：

```
user001::600:100:user:/home/user001:/bin/bash
user002::601:100:user:/home/user002:/bin/bash
user003::602:100:user:/home/user003:/bin/bash
user004::603:100:user:/home/user004:/bin/bash
user005::604:100:user:/home/user005:/bin/bash
user006::605:100:user:/home/user006:/bin/bash
```

（2）以root身份执行命令 `/usr/sbin/newusers` ，从刚创建的用户文件 `user.txt` 中导入数据，创建用户：

```
# newusers < user.txt
```

然后可以执行命令 `vipw` 或 `vi /etc/passwd` 检查 `/etc/passwd` 文件是否已经出现这些用户的数据，并且用户的宿主目录是否已经创建。

（3）执行命令/usr/sbin/pwunconv。

将 `/etc/shadow` 产生的 `shadow` 密码解码，然后回写到 `/etc/passwd` 中，并将 `/etc/shadow` 的 `shadow` 密码栏删掉。这是为了方便下一步的密码转换工作，即先取消 `shadow password` 功能。

```
# pwunconv
```

（4）编辑每个用户的密码对照文件。

范例文件 `passwd.txt` 内容如下：

```
user001:密码
user002:密码
user003:密码
user004:密码
user005:密码
user006:密码
```

（5）以root身份执行命令 `/usr/sbin/chpasswd` 。

创建用户密码， `chpasswd` 会将经过 `/usr/bin/passwd` 命令编码过的密码写入 `/etc/passwd` 的密码栏。

```
# chpasswd < passwd.txt
```

（6）确定密码经编码写入/etc/passwd的密码栏后。

执行命令 `/usr/sbin/pwconv` 将密码编码为 `shadow password`，并将结果写入 `/etc/shadow`。

```
# pwconv
```

这样就完成了大量用户的创建了，之后您可以到/home下检查这些用户宿主目录的权限设置是否都正确，并登录验证用户密码是否正确。

Linux 磁盘管理

Linux磁盘管理好坏管理直接关系到整个系统的性能问题。

Linux磁盘管理常用三个命令为df、du和fdisk。

- df：列出文件系统的整体磁盘使用量
- du：检查磁盘空间使用量
- fdisk：用于磁盘分区

df

df命令参数功能：检查文件系统的磁盘空间占用情况。可以利用该命令来获取硬盘被占用了多少空间，目前还剩下多少空间等信息。

语法：

```
df [-ahikHTm] [目录或文件名]
```

选项与参数：

- -a ：列出所有的文件系统，包括系统特有的 /proc 等文件系统；
- -k ：以 KBytes 的容量显示各文件系统；
- -m ：以 MBytes 的容量显示各文件系统；
- -h ：以人们较易阅读的 GBytes, MBytes, KBytes 等格式自行显示；
- -H ：以 M=1000K 取代 M=1024K 的进位方式；
- -T ：显示文件系统类型, 连同该 partition 的 filesystem 名称 (例如 ext3) 也列出；
- -i ：不用硬盘容量，而以 inode 的数量来显示

实例 1

将系统内所有的文件系统列出来！

```
[root@www ~]# df
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/hdc2         9920624   3823112   5585444  41% /
/dev/hdc3         4956316    141376   4559108   4% /home
/dev/hdc1          101086     11126     84741  12% /boot
tmpfs              371332         0    371332   0% /dev/shm
```

在 Linux 底下如果 df 没有加任何选项，那么默认会将系统内所有的 (不含特殊内存内的文件系统与 swap) 都以 1 Kbytes 的容量来列出来！

实例 2

将容量结果以易读的容量格式显示出来

```
[root@www ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/hdc2             9.5G  3.7G  5.4G  41% /
/dev/hdc3             4.8G  139M  4.4G   4% /home
/dev/hdc1              99M   11M   83M  12% /boot
tmpfs                 363M     0  363M   0% /dev/shm
```

实例 3

将系统内的所有特殊文件格式及名称都列出来

```
[root@www ~]# df -aT
Filesystem    Type 1K-blocks    Used Available Use% Mounted on
/dev/hdc2     ext3   9920624 3823112   5585444  41% /
proc          proc         0       0         0   -  /proc
sysfs        sysfs         0       0         0   -  /sys
devpts      devpts         0       0         0   -  /dev/pts
/dev/hdc3     ext3   4956316  141376   4559108   4% /home
/dev/hdc1     ext3    101086   11126     84741  12% /boot
tmpfs        tmpfs    371332       0    371332   0% /dev/shm
none   binfmt_misc         0       0         0   -  /proc/sys/fs/binfmt_misc
sunrpc  rpc_pipefs         0       0         0   -  /var/lib/nfs/rpc_pipefs
```

实例 4

将 /etc 底下的可用的磁盘容量以易读的容量格式显示

```
[root@www ~]# df -h /etc
Filesystem            Size  Used Avail Use% Mounted on
/dev/hdc2             9.5G  3.7G  5.4G  41% /
```

du

inux du命令也是查看使用空间的，但是与df命令不同的是Linux du命令是对文件和目录磁盘使用的空间的查看，还是和df命令有一些区别的，这里介绍Linux du命令。

语法：

```
du [-ahskm] 文件或目录名称
```

选项与参数：

- -a ：列出所有的文件与目录容量，因为默认仅统计目录底下的文件量而已。
- -h ：以人们较易读的容量格式 (G/M) 显示；

- -s ：列出总量而已，而不列出每个各别的目录占用容量；
- -S ：不包括子目录下的总计，与 -s 有点差别。
- -k ：以 KBytes 列出容量显示；
- -m ：以 MBytes 列出容量显示；

实例 1

列出目前目录下的所有文件容量

```
[root@www ~]# du
8       ./test4     <==每个目录都会列出来
8       ./test2
....中间省略....
12      ./.gconfd   <==包括隐藏文件的目录
220     .           <==这个目录(.)所占用的总量
```

直接输入 du 没有加任何选项时，则 du 会分析当前所在目录的文件与目录所占用的硬盘空间。

实例 2

将文件的容量也列出来

```
[root@www ~]# du -a
12      ./install.log.syslog   <==有文件的列表了
8       ./.bash_logout
8       ./test4
8       ./test2
....中间省略....
12      ./.gconfd
220     .
```

实例 3

检查根目录底下每个目录所占用的容量

```
[root@www ~]# du -sm /*
7       /bin
6       /boot
.....中间省略....
0       /proc
.....中间省略....
1       /tmp
3859    /usr     <==系统初期最大就是他了啦！
77      /var
```

通配符 * 来代表每个目录。

与 df 不一样的是，du 这个命令其实会直接到文件系统内去搜寻所有的文件数据。

fdisk

fdisk 是 Linux 的磁盘分区表操作工具。

语法：

```
fdisk [-l] 装置名称
```

选项与参数：

- -l ：输出后面接的装置所有的分区内容。若仅有 fdisk -l 时， 则系统将会把整个系统内能够搜寻到的装置的分区均列出来。

实例 1

列出所有分区信息

```
[root@AY120919111755c246621 tmp]# fdisk -l

Disk /dev/xvda: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *           1        2550    20480000   83  Linux
/dev/xvda2            2550        2611      490496   82  Linux swap / Solaris

Disk /dev/xvdb: 21.5 GB, 21474836480 bytes
255 heads, 63 sectors/track, 2610 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x56f40944

    Device Boot      Start         End      Blocks   Id  System
/dev/xvdb2               1        2610    20964793+  83  Linux
```

实例 2

找出你系统中的根目录所在磁盘，并查阅该硬盘内的相关信息

```
[root@www ~]# df /            <==注意：重点在找出磁盘文件名而已
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hdc2              9920624   3823168   5585388  41% /

[root@www ~]# fdisk /dev/hdc  <==仔细看，不要加上数字喔！
The number of cylinders for this disk is set to 5005.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help):     <==等待你的输入！
```

输入 m 后，就会看到底下这些命令介绍

```
Command (m for help): m   <== 输入 m 后，就会看到底下这些命令介绍
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition            <==删除一个partition
   l   list known partition types
   m   print this menu
   n   add a new partition           <==新增一个partition
   o   create a new empty DOS partition table
   p   print the partition table     <==在屏幕上显示分割表
   q   quit without saving changes   <==不储存离开fdisk程序
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit  <==将刚刚的动作写入分割表
   x   extra functionality (experts only)
```

离开 fdisk 时按下 `q`，那么所有的动作都不会生效！相反的， 按下 `w` 就是动作生效的意思。

```
Command (m for help): p  <== 这里可以输出目前磁盘的状态

Disk /dev/hdc: 41.1 GB, 41174138880 bytes        <==这个磁盘的文件名与容量
255 heads, 63 sectors/track, 5005 cylinders      <==磁头、扇区与磁柱大小
Units = cylinders of 16065 * 512 = 8225280 bytes <==每个磁柱的大小

   Device Boot      Start         End      Blocks   Id  System
/dev/hdc1   *           1          13      104391   83  Linux
/dev/hdc2              14        1288    10241437+  83  Linux
/dev/hdc3            1289        1925     5116702+  83  Linux
/dev/hdc4            1926        5005    24740100    5  Extended
/dev/hdc5            1926        2052     1020096   82  Linux swap / Solaris
# 装置文件名 启动区否 开始磁柱    结束磁柱  1K大小容量 磁盘分区槽内的系统

Command (m for help): q
```

想要不储存离开吗？按下 q 就对了！不要随便按 w 啊！

使用 `p` 可以列出目前这颗磁盘的分割表信息，这个信息的上半部在显示整体磁盘的状态。

磁盘格式化

磁盘分割完毕后自然就是要进行文件系统的格式化，格式化的命令非常的简单，使用 `mkfs` （make filesystem） 命令。

语法：

```
mkfs [-t 文件系统格式] 装置文件名
```

选项与参数：

- -t ：可以接文件系统格式，例如 ext3, ext2, vfat 等(系统有支持才会生效)

实例 1

查看 mkfs 支持的文件格式

```
[root@www ~]# mkfs[tab][tab]
mkfs         mkfs.cramfs  mkfs.ext2    mkfs.ext3    mkfs.msdos   mkfs.vfat
```

按下两个[tab]，会发现 mkfs 支持的文件格式如上所示。

实例 2

将分区 /dev/hdc6（可指定你自己的分区） 格式化为 ext3 文件系统：

```
[root@www ~]# mkfs -t ext3 /dev/hdc6
mke2fs 1.39 (29-May-2006)
Filesystem label=                <==这里指的是分割槽的名称(label)
OS type: Linux
Block size=4096 (log=2)          <==block 的大小配置为 4K
Fragment size=4096 (log=2)
251392 inodes, 502023 blocks     <==由此配置决定的inode/block数量
25101 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=515899392
16 block groups
32768 blocks per group, 32768 fragments per group
15712 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Writing inode tables: done
Creating journal (8192 blocks): done <==有日志记录
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
# 这样就创建起来我们所需要的 Ext3 文件系统了！简单明了！
```

磁盘检验

fsck（file system check）用来检查和维护不一致的文件系统。

若系统掉电或磁盘发生问题，可利用fsck命令对文件系统进行检查。

语法：

```
fsck [-t 文件系统] [-ACay] 装置名称
```

选项与参数：

- -t : 给定档案系统的型式，若在 /etc/fstab 中已有定义或 kernel 本身已支援的则不需加上此参数
- -s : 依序一个一个地执行 fsck 的指令来检查
- -A : 对/etc/fstab 中所有列出来的 分区（partition）做检查
- -C : 显示完整的检查进度
- -d : 打印出 e2fsck 的 debug 结果
- -p : 同时有 -A 条件时，同时有多个 fsck 的检查一起执行
- -R : 同时有 -A 条件时，省略 / 不检查
- -V : 详细显示模式
- -a : 如果检查有错则自动修复
- -r : 如果检查有错则由使用者回答是否修复
- -y : 选项指定检测每个文件是自动输入yes，在不确定那些是不正常的时候，可以执行 # fsck -y 全部检查修复。

实例 1

查看系统有多少文件系统支持的 fsck 命令：

```
[root@www ~]# fsck[tab][tab]
fsck         fsck.cramfs  fsck.ext2    fsck.ext3    fsck.msdos   fsck.vfat
```

实例 2

强制检测 /dev/hdc6 分区:

```
[root@www ~]# fsck -C -f -t ext3 /dev/hdc6
fsck 1.39 (29-May-2006)
e2fsck 1.39 (29-May-2006)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
vbird_logical: 11/251968 files (9.1% non-contiguous), 36926/1004046 blocks
```

如果没有加上 -f 的选项，则由于这个文件系统不曾出现问题，检查的经过非常快速！若加上 -f 强制检查，才会一项一项的显示过程。

磁盘挂载与卸除

Linux 的磁盘挂载使用 `mount` 命令，卸载使用 `umount` 命令。

磁盘挂载语法：

```
mount [-t 文件系统] [-L Label名] [-o 额外选项] [-n]  装置文件名  挂载点
```

实例 1

用默认的方式，将刚刚创建的 /dev/hdc6 挂载到 /mnt/hdc6 上面！

```
[root@www ~]# mkdir /mnt/hdc6
[root@www ~]# mount /dev/hdc6 /mnt/hdc6
[root@www ~]# df
Filesystem           1K-blocks      Used Available Use% Mounted on
.....中间省略.....
/dev/hdc6              1976312     42072   1833836   3% /mnt/hdc6
```

磁盘卸载命令 `umount` 语法：

```
umount [-fn] 装置文件名或挂载点
```

选项与参数：

- -f ：强制卸除！可用在类似网络文件系统 (NFS) 无法读取到的情况下；
- -n ：不升级 /etc/mtab 情况下卸除。

卸载/dev/hdc6

```
[root@www ~]# umount /dev/hdc6
```

Linux vi/vim

所有的 Unix Like 系统都会内建 vi 文书编辑器，其他的文书编辑器则不一定会存在。

但是目前我们使用比较多的是 vim 编辑器。

vim 具有程序编辑的能力，可以主动的以字体颜色辨别语法的正确性，方便程序设计。

什么是 vim？

Vim是从 vi 发展出来的一个文本编辑器。代码补完、编译及错误跳转等方便编程的功能特别丰富，在程序员中被广泛使用。

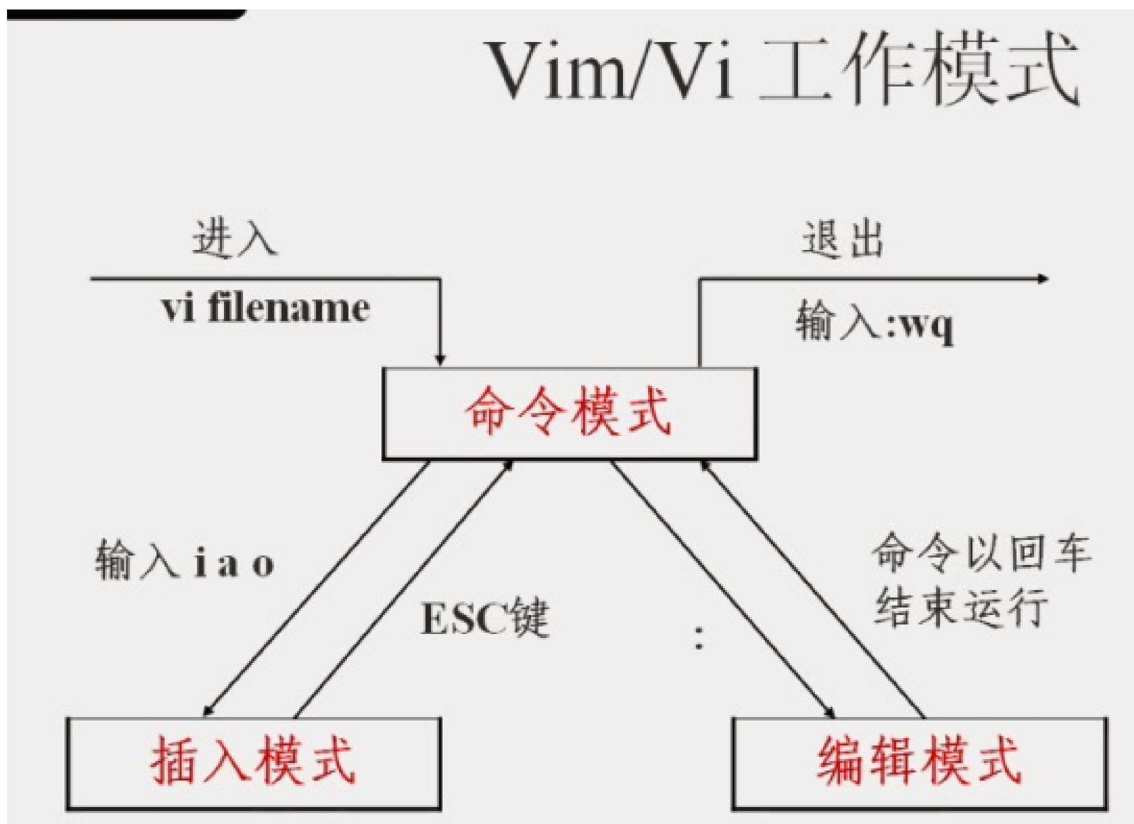
简单的来说， vi 是老式的字处理器，不过功能已经很齐全了，但是还是有可以进步的地方。vim 则可以说是程序开发者的一项很好用的工具。 连 vim 的官方网站 (http://www.vim.org) 自己也说 vim 是一个程序开发工具而不是文字处理软件。

vi/vim 的使用

基本上 vi/vim 共分为三种模式，分别是一般模式、编辑模式与指令列命令模式。 这三种模式的作用分别是：

- 一般模式：
  以 vi 打开一个档案就直接进入一般模式了(这是默认的模式)。在这个模式中， 你可以使用『上下左右』按键来移动光标，你可以使用『删除字符』或『删除整行』来处理档案内容， 也可以使用『复制、贴上』来处理你的文件数据。
- 编辑模式：
  在一般模式中可以进行删除、复制、贴上等等的动作，但是却无法编辑文件内容的！ 要等到你按下『i, I, o, O, a, A, r, R』等任何一个字母之后才会进入编辑模式。注意了！通常在 Linux 中，按下这些按键时，在画面的左下方会出现『INSERT 或 REPLACE 』的字样，此时才可以进行编辑。而如果要回到一般模式时， 则必须要按下『Esc』这个按键即可退出编辑模式。
- 指令列命令模式：
  在一般模式当中，输入『 : / ? 』三个中的任何一个按钮，就可以将光标移动到最底下那一行。在这个模式当中， 可以提供你『搜寻资料』的动作，而读取、存盘、大量取代字符、离开 vi 、显示行号等等的动作则是在此模式中达成的！

简单的说，我们可以将这三个模式想成底下的图标来表示：



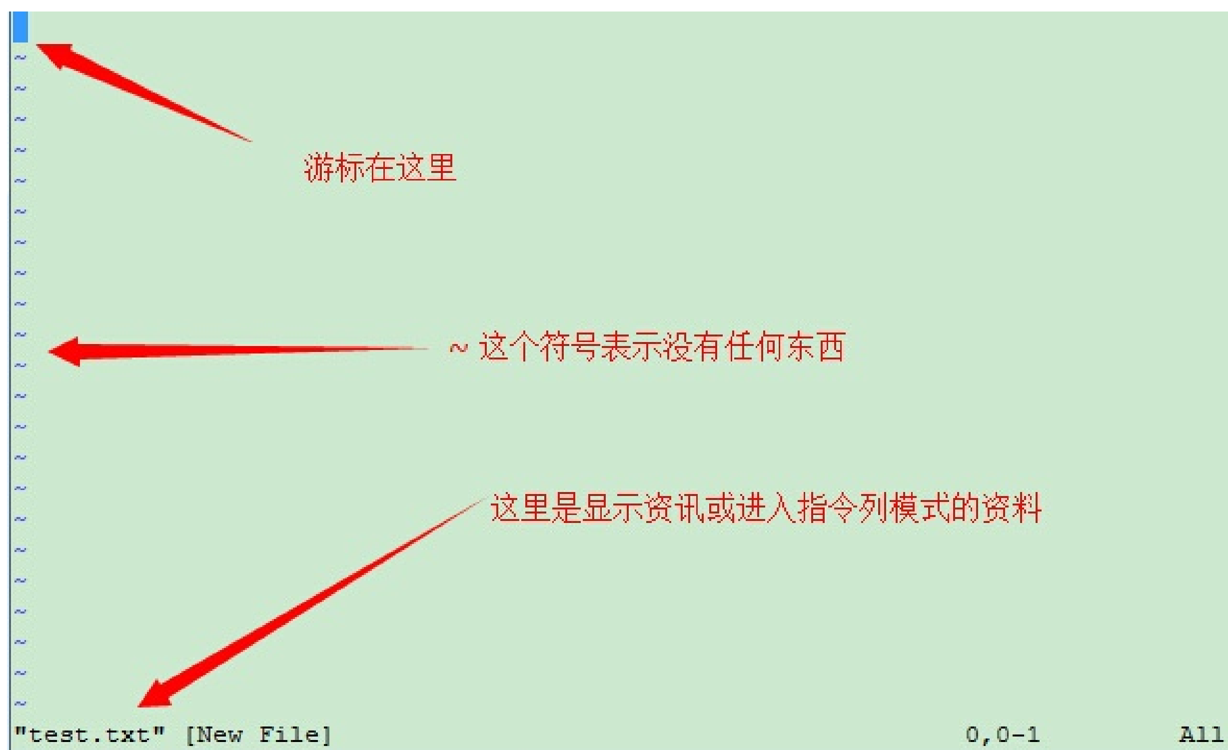
vi/vim 使用实例

使用 vi/vim 进入一般模式

如果你想要使用 vi 来建立一个名为 test.txt 的文件时，你可以这样做：

```
[root@www ~]# vi test.txt
```

直接输入 **vi** 文件名 就能够进入 vi 的一般模式了。请注意，记得 vi 后面一定要加文件名，不管该文件存在与否！



按下 i 进入编辑模式，开始编辑文字

在一般模式之中，只要按下 i, o, a 等字符就可以进入编辑模式了！

在编辑模式当中，你可以发现在左下角状态栏中会出现 –INSERT- 的字样，那就是可以输入任意字符的提示。

这个时候，键盘上除了 [Esc] 这个按键之外，其他的按键都可以视作为一般的输入按钮了，所以你可以进行任何的编辑。



按下 [ESC] 按钮回到一般模式

好了，假设我已经按照上面的样式给他编辑完毕了，那么应该要如何退出呢？是的！没错！就是给他按下 [Esc] 这个按钮即可！马上你就会发现画面左下角的 – INSERT – 不见了！

在一般模式中按下 :wq 储存后离开 vi

OK，我们要存档了，存盘并离开的指令很简单，输入『:wq』即可保存离开！



OK! 这样我们就成功创建了一个 test.txt 的文件。是不是很简单。

vi/vim 按键说明

除了上面简易范例的 i, [Esc], :wq 之外，其实 vim 还有非常多的按键可以使用。

第一部份：一般模式可用的按钮说明，光标移动、复制贴上、搜寻取代等

移动光标的方法	
h 或 向左箭头键(←)	光标向左移动一个字符
j 或 向下箭头键(↓)	光标向下移动一个字符
k 或 向上箭头键(↑)	光标向上移动一个字符
l 或 向右箭头键(→)	光标向右移动一个字符
如果你将右手放在键盘上的话，你会发现 hjkl 是排列在一起的，因此可以使用这四个按钮来移动光标。 如果想要进行	

多次移动的话，例如向下移动 30 行，可以使用 "30j" 或 "30↓" 的组合按键，亦即加上想要进行的次数(数字)后，按下动作即可！	
[Ctrl] + [f]	屏幕『向下』移动一页，相当于 [Page Down] 按键 (常用)
[Ctrl] + [b]	屏幕『向上』移动一页，相当于 [Page Up] 按键 (常用)
[Ctrl] + [d]	屏幕『向下』移动半页
[Ctrl] + [u]	屏幕『向上』移动半页
+	光标移动到非空格符的下一列
-	光标移动到非空格符的上一列
n	那个 n 表示『数字』，例如 20 。按下数字后再按空格键，光标会向右移动这一行的 n 个字符。例如 20 则光标会向后面移动 20 个字符距离。
0 或功能键[Home]	这是数字『 0 』：移动到这一行的最前面字符处 (常用)
$ 或功能键[End]	移动到这一行的最后面字符处(常用)
H	光标移动到这个屏幕的最上方那一行的第一个字符
M	光标移动到这个屏幕的中央那一行的第一个字符
L	光标移动到这个屏幕的最下方那一行的第一个字符
G	移动到这个档案的最后一行(常用)
nG	n 为数字。移动到这个档案的第 n 行。例如 20G 则会移动到这个档案的第 20 行(可配合 :set nu)
gg	移动到这个档案的第一行，相当于 1G 啊！(常用)
n	n 为数字。光标向下移动 n 行(常用)
搜寻与取代	
/word	向光标之下寻找一个名称为 word 的字符串。例如要在档案内搜寻 vbird 这个字符串，就输入 /vbird 即可！ (常用)
?word	向光标之上寻找一个字符串名称为 word 的字符串。
	这个 n 是英文按键。代表重复前一个搜寻的

n	动作。举例来说， 如果刚刚我们执行 /vbird 去向下搜寻 vbird 这个字符串，则按下 n 后，会向下继续搜寻下一个名称为 vbird 的字符串。如果是执行 ?vbird 的话，那么按下 n 则会向上继续搜寻名称为 vbird 的字符串！
N	这个 N 是英文按键。与 n 刚好相反，为『反向』进行前一个搜寻动作。 例如 /vbird 后，按下 N 则表示『向上』搜寻 vbird 。
使用 /word 配合 n 及 N 是非常有帮助的！可以让你重复的找到一些你搜寻的关键词！	
:n1,n2s/word1/word2/g	n1 与 n2 为数字。在第 n1 与 n2 行之间寻找 word1 这个字符串，并将该字符串取代为 word2 ！举例来说，在 100 到 200 行之间搜寻 vbird 并取代为 VBIRD 则：『:100,200s/vbird/VBIRD/g』。(常用)
:1,$s/word1/word2/g	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！(常用)
:1,$s/word1/word2/gc	从第一行到最后一行寻找 word1 字符串，并将该字符串取代为 word2 ！且在取代前显示提示字符给用户确认 (confirm) 是否需要取代！(常用)
x, X	在一行字当中，x 为向后删除一个字符 (相当于 [del] 按键)， X 为向前删除一个字符(相当于 [backspace] 亦即是退格键) (常用)
nx	n 为数字，连续向后删除 n 个字符。举例来说，我要连续删除 10 个字符， 『10x』。
dd	删除游标所在的那一整列(常用)
ndd	n 为数字。删除光标所在的向下 n 列，例如 20dd 则是删除 20 列 (常用)
d1G	删除光标所在到第一行的所有数据
dG	删除光标所在到最后一行的所有数据
d$	删除游标所在处，到该行的最后一个字符
d0	那个是数字的 0 ，删除游标所在处，到该行的最前面一个字符
yy	复制游标所在的那一行(常用)
nyy	n 为数字。复制光标所在的向下 n 列，例如 20yy 则是复制 20 列(常用)
y1G	复制游标所在列到第一列的所有数据
yG	复制游标所在列到最后一列的所有数据

y0	复制光标所在的那个字符到该行行首的所有数据
y$	复制光标所在的那个字符到该行行尾的所有数据
p, P	p 为将已复制的数据在光标下一行贴上，P 则为贴在游标上一行！ 举例来说，我目前光标在第 20 行，且已经复制了 10 行数据。则按下 p 后， 那 10 行数据会贴在原本的 20 行之后，亦即由 21 行开始贴。但如果是按下 P 呢？ 那么原本的第 20 行会被推到变成 30 行。 (常用)
J	将光标所在列与下一列的数据结合成同一列
c	重复删除多个数据，例如向下删除 10 行，[10cj]
u	复原前一个动作。(常用)
[Ctrl]+r	重做上一个动作。(常用)
这个 u 与 [Ctrl]+r 是很常用的指令！一个是复原，另一个则是重做一次～ 利用这两个功能按键，你的编辑，嘿嘿！很快乐的啦！	
.	不要怀疑！这就是小数点！意思是重复前一个动作的意思。 如果你想要重复删除、重复贴上等等动作，按下小数点『.』就好了！ (常用)

第二部份：一般模式切换到编辑模式的可用的按钮说明

进入插入或取代的编辑模式	
i, I	进入插入模式(Insert mode)： i 为『从目前光标所在处插入』， I 为『在目前所在行的第一个非空格符处开始插入』。 (常用)
a, A	进入插入模式(Insert mode)： a 为『从目前光标所在的下一个字符处开始插入』， A 为『从光标所在行的最后一个字符处开始插入』。(常用)
o, O	进入插入模式(Insert mode)： 这是英文字母 o 的大小写。o 为『在目前光标所在的下一行处插入新的一行』； O 为在目前光标所在处的上一行插入新的一行！(常用)
r, R	进入取代模式(Replace mode)：r 只会取代光标所在的那一个字符一次；R会一直取代光标所在的文字，直到按下 ESC 为止；(常用)
上面这些按键中，在 vi 画面的左下角处会出现『--INSERT--』或『--REPLACE--』的字样。 由名称就知道该动作了吧！！特别注意的是，我们上面也提过了，你想要在档案里面输入字符时， 一定要在左下角处看到 INSERT 或 REPLACE 才能输入喔！	
[Esc]	退出编辑模式，回到一般模式中(常用)

第三部份：一般模式切换到指令列模式的可用的按钮说明

指令列的储存、离开等指令	
:w	将编辑的数据写入硬盘档案中(常用)
:w!	若文件属性为『只读』时，强制写入该档案。不过，到底能不能写入， 还是跟你对该档案的档案权限有关啊！
:q	离开 vi (常用)
:q!	若曾修改过档案，又不想储存，使用！为强制离开不储存档案。
注意一下啊，那个惊叹号 (!) 在 vi 当中，常常具有『强制』的意思～	
:wq	储存后离开，若为 :wq! 则为强制储存后离开 (常用)
ZZ	这是大写的 Z 喔！若档案没有更动，则不储存离开，若档案已经被更动过，则储存后离开！
:w [filename]	将编辑的数据储存成另一个档案（类似另存新档）
:r [filename]	在编辑的数据中，读入另一个档案的数据。亦即将『filename』 这个档案内容加到游标所在行后面
:n1,n2 w [filename]	将 n1 到 n2 的内容储存成 filename 这个档案。
:! command	暂时离开 vi 到指令列模式下执行 command 的显示结果！例如 『:! ls /home』即可在 vi 当中察看 /home 底下以 ls 输出的档案信息！
vim 环境的变更	
:set nu	显示行号，设定之后，会在每一行的前缀显示该行的行号
:set nonu	与 set nu 相反，为取消行号！

特别注意，在 vi/vim 中，数字是很有意义的！数字通常代表重复做几次的意思！ 也有可能是代表去到第几个什么什么的意思。

举例来说，要删除 50 行，则是用 『50dd』 对吧！ 数字加在动作之前，如我要向下移动 20 行呢？那就是『20j』或者是『20↓』即可。

linux yum 命令

yum（ Yellow dog Updater, Modified）是一个在Fedora和RedHat以及SUSE中的Shell前端软件包管理器。

基於RPM包管理，能够从指定的服务器自动下载RPM包并且安装，可以自动处理依赖性关系，并且一次安装所有依赖的软体包，无须繁琐地一次次下载、安装。

yum提供了查找、安装、删除某一个、一组甚至全部软件包的命令，而且命令简洁而又好记。

yum 语法

```
yum [options] [command] [package ...]
```

- **options**：可选，选项包括-h（帮助），-y（当安装过程提示选择全部为"yes"），-q（不显示安装的过程）等等。
- **command**：要进行的操作。
- **package**操作的对象。

yum常用命令

- 1.列出所有可更新的软件清单命令：yum check-update
- 2.更新所有软件命令：yum update
- 3.仅安装指定的软件命令：yum install <package_name>
- 4.仅更新指定的软件命令：yum update <package_name>
- 5.列出所有可安裝的软件清单命令：yum list
- 6.删除软件包命令：yum remove <package_name>
- 7.查找软件包 命令：yum search <keyword>
- 8.清除缓存命令:
  - yum clean packages: 清除缓存目录下的软件包
  - yum clean headers: 清除缓存目录下的 headers
  - yum clean oldheaders: 清除缓存目录下旧的 headers
  - yum clean, yum clean all (= yum clean packages; yum clean oldheaders) :清除缓存目录下的软件包及旧的headers

实例 1

安装 pam-devel

```
[root@www ~]# yum install pam-devel
Setting up Install Process
Parsing package install arguments
Resolving Dependencies  <==先检查软件的属性相依问题
--> Running transaction check
---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be updated
--> Processing Dependency: pam = 0.99.6.2-4.el5 for package: pam-devel
--> Running transaction check
---> Package pam.i386 0:0.99.6.2-4.el5 set to be updated
filelists.xml.gz          100% |=========================| 1.6 MB    00:05
filelists.xml.gz          100% |=========================| 138 kB    00:00
-> Finished Dependency Resolution
……(省略)
```

实例 2

移除 pam-devel

```
[root@www ~]# yum remove pam-devel
Setting up Remove Process
Resolving Dependencies  <==同样的，先解决属性相依的问题
--> Running transaction check
---> Package pam-devel.i386 0:0.99.6.2-4.el5 set to be erased
--> Finished Dependency Resolution

Dependencies Resolved

=============================================================================
 Package                 Arch       Version          Repository        Size
=============================================================================
Removing:
 pam-devel               i386       0.99.6.2-4.el5   installed         495 k

Transaction Summary
=============================================================================
Install      0 Package(s)
Update       0 Package(s)
Remove       1 Package(s)  <==还好，并没有属性相依的问题，单纯移除一个软件

Is this ok [y/N]: y
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Erasing    : pam-devel                    ######################### [1/1]

Removed: pam-devel.i386 0:0.99.6.2-4.el5
Complete!
```

实例 3

利用 yum 的功能，找出以 pam 为开头的软件名称有哪些？

```
[root@www ~]# yum list pam*
Installed Packages
pam.i386                  0.99.6.2-3.27.el5      installed
pam_ccreds.i386           3-5                    installed
pam_krb5.i386             2.2.14-1               installed
pam_passwdqc.i386         1.0.2-1.2.2            installed
pam_pkcs11.i386           0.5.3-23               installed
pam_smb.i386              1.1.7-7.2.1            installed
Available Packages <==底下则是『可升级』的或『未安装』的
pam.i386                  0.99.6.2-4.el5         base
pam-devel.i386            0.99.6.2-4.el5         base
pam_krb5.i386             2.2.14-10              base
```

国内 yum 源

网易（163）yum源是国内最好的yum源之一 ，无论是速度还是软件版本，都非常的不错。

将yum源设置为163 yum，可以提升软件包安装和更新的速度，同时避免一些常见软件版本无法找到。

安装步骤

首先备份/etc/yum.repos.d/CentOS-Base.repo

```
mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup
```

下载对应版本repo文件, 放入/etc/yum.repos.d/(操作前请做好相应备份)

- CentOS5 ：http://mirrors.163.com/.help/CentOS5-Base-163.repo
- CentOS6 ：http://mirrors.163.com/.help/CentOS6-Base-163.repo

运行以下命令生成缓存

```
yum clean all
yum makecache
```

除了网易之外，国内还有其他不错的yum源，比如中科大和搜狐。

中科大的yum源，安装方法查看：https://lug.ustc.edu.cn/wiki/mirrors/help/centos

sohu的yum源安装方法查看: http://mirrors.sohu.com/help/centos.html

Linux命令大全

Linux命令大全 - 文件管理

cat	chattr	chgrp	chmod
chown	cksum	cmp	diff
diffstat	file	find	git
gitview	indent	cut	ln
less	locate	lsattr	mattrib
mc	mdel	mdir	mktemp
more	mmove	mread	mren
mtools	mtoolstest	mv	od
paste	patch	rcp	rm
slocate	split	tee	tmpwatch
touch	umask	which	cp
whereis	mcopy	mshowfat	rhmask
scp	awk		

Linux cat命令

命令：cat

cat命令用于把档案串连接后传到基本输出（萤幕或加 > fileName 到另一个档案）

使用权限

所有使用者

语法格式

```
cat [-AbeEnstTuv] [--help] [--version] fileName
```

参数说明：

-n 或 --number 由 1 开始对所有输出的行数编号

-b 或 --number-nonblank 和 -n 相似，只不过对于空白行不编号

-s 或 --squeeze-blank 当遇到有连续两行以上的空白行，就代换为一行的空白行

-v 或 --show-nonprinting

实例：

把 textfile1 的档案内容加上行号后输入 textfile2 这个档案里

```
cat -n textfile1 &gt; textfile2
```

把 textfile1 和 textfile2 的档案内容加上行号（空白行不加）之后将内容附加到 textfile3 里。

```
cat -b textfile1 textfile2 &gt;&gt; textfile3
```

清空/etc/test.txt档案内容

```
cat /dev/null &gt; /etc/test.txt
```

cat 也可以用来制作镜像文件。例如要制作软碟的像文件，将软碟放好后打

```
cat /dev/fd0 &gt; OUTFILE
```

相反的，如果想把 image file 写到软碟，请打

```
cat IMG_FILE &gt; /dev/fd0
```

注：

- 1. OUTFILE 指输出的镜像文件名。
- 2. IMG_FILE 指镜像文件。
- 3. 若从镜像文件写回 device 时，device 容量需与相当。
- 4. 通常用在制作开机磁片。

Linux chattr命令

Linux chattr命令用于改变文件属性。

这项指令可改变存放在ext2文件系统上的文件或目录属性，这些属性共有以下8种模式：

1. a：让文件或目录仅供附加用途。
2. b：不更新文件或目录的最后存取时间。
3. c：将文件或目录压缩后存放。
4. d：将文件或目录排除在倾倒操作之外。
5. i：不得任意更动文件或目录。
6. s：保密性删除文件或目录。
7. S：即时更新文件或目录。
8. u：预防以外删除。

语法

```
chattr [-RV][-v<版本编号>][+/-/=<属性>][文件或目录...]
```

参数

-R 递归处理，将指定目录下的所有文件及子目录一并处理。

-v<版本编号> 设置文件或目录版本。

-V 显示指令执行过程。

+<属性> 开启文件或目录的该项属性。

-<属性> 关闭文件或目录的该项属性。

=<属性> 指定文件或目录的该项属性。

实例

用chattr命令防止系统中某个关键文件被修改：

```
chattr +i /etc/resolv.conf
```

```
lsattr /etc/resolv.conf
```

会显示如下属性

```
----i-------- /etc/resolv.conf
```

让某个文件只能往里面追加数据，但不能删除，适用于各种日志文件：

```
chattr +a /var/log/messages
```

Linux chgrp命令

Linux chgrp命令用于变更文件或目录的所属群组。

在UNIX系统家族里，文件或目录权限的掌控以拥有者及所属群组来管理。您可以使用chgrp指令去变更文件与目录的所属群组，设置方式采用群组名称或群组识别码皆可。

语法

```
chgrp [-cfhRv][--help][--version][所属群组][文件或目录...] 或 chgrp [-cfhRv][--help][--refer
```

参数说明

-c或--changes 效果类似"-v"参数，但仅回报更改的部分。

-f或--quiet或--silent 不显示错误信息。

-h或--no-dereference 只对符号连接的文件作修改，而不更动其他任何相关文件。

-R或--recursive 递归处理，将指定目录下的所有文件及子目录一并处理。

-v或--verbose 显示指令执行过程。

--help 在线帮助。

--reference=<参考文件或目录> 把指定文件或目录的所属群组全部设成和参考文件或目录的所属群组相同。

--version 显示版本信息。

实例

实例1：改变文件的群组属性：

```
chgrp -v bin log2012.log
```

输出：

```
[root@localhost test]# ll
---xrw-r-- 1 root root 302108 11-13 06:03 log2012.log
[root@localhost test]# chgrp -v bin log2012.log
```

"log2012.log" 的所属组已更改为 bin

```
[root@localhost test]# ll
---xrw-r-- 1 root bin  302108 11-13 06:03 log2012.log
```

说明： 将log2012.log文件由root群组改为bin群组

实例2：根据指定文件改变文件的群组属性

```
chgrp --reference=log2012.log log2013.log
```

输出：

```
[root@localhost test]# ll
---xrw-r-- 1 root bin  302108 11-13 06:03 log2012.log
-rw-r--r-- 1 root root     61 11-13 06:03 log2013.log
[root@localhost test]#  chgrp --reference=log2012.log log2013.log
[root@localhost test]# ll
---xrw-r-- 1 root bin  302108 11-13 06:03 log2012.log
-rw-r--r-- 1 root bin      61 11-13 06:03 log2013.log
```

说明： 改变文件log2013.log 的群组属性，使得文件log2013.log的群组属性和参考文件log2012.log的群组属性相同

Linux chmod命令

Linux/Unix 的文件调用权限分为三级 : 文件拥有者、群组、其他。利用 chmod 可以藉以控制文件如何被他人所调用。

使用权限 : 所有使用者

语法

```
chmod [-cfvR] [--help] [--version] mode file...
```

参数说明

- mode : 权限设定字串，格式如下 :

  ```
  [ugoa...][[+-=][rwxX]...][,...]
  ```

  其中 :

  - u 表示该文件的拥有者，g 表示与该文件的拥有者属于同一个群体(group)者，o 表示其他以外的人，a 表示这三者皆是。
  -
    - 表示增加权限、- 表示取消权限、= 表示唯一设定权限。
  - r 表示可读取，w 表示可写入，x 表示可执行，X 表示只有当该文件是个子目录或者该文件已经被设定过为可执行。 *-c :* 若该文件权限确实已经更改，才显示其更改动作 -f : 若该文件权限无法被更改也不要显示错误讯息 *-v :* 显示权限变更的详细资料 -R : 对目前目录下的所有文件与子目录进行相同的权限变更(即以递回的方式逐个变更) *--help :* 显示辅助说明 --version : 显示版本

实例

将文件 file1.txt 设为所有人皆可读取 :

```
chmod ugo+r file1.txt
```

将文件 file1.txt 设为所有人皆可读取 :

```
chmod a+r file1.txt
```

将文件 file1.txt 与 file2.txt 设为该文件拥有者，与其所属同一个群体者可写入，但其他以外的人则不可写入：

```
chmod ug+w,o-w file1.txt file2.txt
```

将 ex1.py 设定为只有该文件拥有者可以执行：

```
chmod u+x ex1.py
```

将目前目录下的所有文件与子目录皆设为任何人可读取：

```
chmod -R a+r *
```

此外chmod也可以用数字来表示权限如：

```
chmod 777 file
```

语法为：

```
chmod abc file
```

其中a,b,c各为一个数字，分别表示User、Group、及Other的权限。

r=4，w=2，x=1

- 若要rwx属性则4+2+1=7；
- 若要rw-属性则4+2=6；
- 若要r-x属性则4+1=5。

```
chmod a=rwx file
```

和

```
chmod 777 file
```

效果相同

```
chmod ug=rwx,o=x file
```

和

```
chmod 771 file
```

效果相同

若用chmod 4755 filename可使此程序具有root的权限

Linux chown命令

Linux/Unix 是多人多工操作系统，所有的文件皆有拥有者。利用 chown 将指定文件的拥有者改为指定的用户或组，用户可以是用户名或者用户ID；组可以是组名或者组ID；文件是以空格分开的要改变权限的文件列表，支持通配符。 。

一般来说，这个指令只有是由系统管理者(root)所使用，一般使用者没有权限可以改变别人的文件拥有者，也没有权限可以自己的文件拥有者改设为别人。只有系统管理者(root)才有这样的权限。

使用权限 : root

语法

```
chmod [-cfhvR] [--help] [--version] user[:group] file...
```

参数 :

- user : 新的文件拥有者的使用者 ID
- group : 新的文件拥有者的使用者群体(group)
- -c : 若该文件拥有者确实已经更改，才显示其更改动作
- -f : 若该文件拥有者无法被更改也不要显示错误讯息
- -h : 只对于连结(link)进行变更，而非该 link 真正指向的文件
- -v : 显示拥有者变更的详细资料
- -R : 对目前目录下的所有文件与子目录进行相同的拥有者变更(即以递回的方式逐个变更)
- --help : 显示辅助说明
- --version : 显示版本

实例

将文件 file1.txt 的拥有者设为 users 群体的使用者 jessie :

```
chown jessie:users file1.txt
```

将目前目录下的所有文件与子目录的拥有者皆设为 users 群体的使用者 lamport :

```
chmod -R lamport:users *
```

Linux cksum命令

Linux cksum命令用于检查文件的CRC是否正确。确保文件从一个系统传输到另一个系统的过程中不被损坏。

CRC是一种排错检查方式，该校验法的标准由CCITT所指定，至少可检测到99.998%的已知错误。

指定文件交由指令"cksum"进行校验后，该指令会返回校验结果供用户核对文件是否正确无误。若不指定任何文件名称或是所给予的文件名为"-"，则指令"cksum"会从标准输入设备中读取数据。

语法

```
cksum [--help][--version][文件...]
```

参数：

- --help：在线帮助。
- --version：显示版本信息。
- 文件…:需要进行检查的文件路径

实例

使用指令"cksum"计算文件"testfile1"的完整性，输入如下命令：

```
$ cksum testfile1
```

以上命令执行后，将输出校验码等相关的信息，具体输出信息如下所示：

```
1263453430 78 testfile1         //输出信息
```

上面的输出信息中，"1263453430"表示校验码，"78"表示字节数。

注意：如果文件中有任何字符被修改，都将改变计算后CRC校验码的值。

Linux cmp命令

Linux cmp命令用于比较两个文件是否有差异。

当相互比较的两个文件完全一样时，则该指令不会显示任何信息。若发现有所差异，预设会标示出第一个不同之处的字符和列数编号。若不指定任何文件名称或是所给予的文件名为"-"，则cmp指令会从标准输入设备读取数据。

语法

```
cmp [-clsv][-i <字符数目>][--help][第一个文件][第二个文件]
```

参数：

- -c或--print-chars　除了标明差异处的十进制字码之外，一并显示该字符所对应字符。
- -i<字符数目>或--ignore-initial=<字符数目>　指定一个数目。
- -l或--verbose　标示出所有不一样的地方。
- -s或--quiet或--silent　不显示错误信息。
- -v或--version　显示版本信息。
- --help　在线帮助。

实例

要确定两个文件是否相同，请输入：

```
cmp prog.o.bak prog.o
```

这比较 prog.o.bak 和 prog.o。如果文件相同，则不显示消息。如果文件不同，则显示第一个不同的位置；例如：

```
prog.o.bak prog.o differ: char 4, line 1
```

如果显示消息 cmp: EOF on prog.o.bak，则 prog.o 的第一部分与 prog.o.bak 相同，但在 prog.o 中还有其他数据。

Linux diff命令

Linux diff命令用于比较文件的差异。

diff以逐行的方式，比较文本文件的异同处。所是指定要比较目录，则diff会比较目录中相同文件名的文件，但不会比较其中子目录。

语法

```
diff [-abBcdefHilnNpPqrstTuvwy][-<行数>][-C <行数>][-D <巨集名称>][-I <字符或字符串>][-S <文件
```

参数：

- -<行数> 指定要显示多少行的文本。此参数必须与-c或-u参数一并使用。 *-a或--text diff*预设只会逐行比较文本文件。 -b或--ignore-space-change 不检查空格字符的不同。
- -B或--ignore-blank-lines 不检查空白行。
- -c 显示全部内文，并标出不同之处。
- -C<行数>或--context<行数> 与执行"-c-<行数>"指令相同。
- -d或--minimal 使用不同的演算法，以较小的单位来做比较。
- -D<巨集名称>或ifdef<巨集名称> 此参数的输出格式可用于前置处理器巨集。
- -e或--ed 此参数的输出格式可用于ed的script文件。
- -f或-forward-ed 输出的格式类似ed的script文件，但按照原来文件的顺序来显示不同处。
- -H或--speed-large-files 比较大文件时，可加快速度。
- -I<字符或字符串>或--ignore-matching-lines<字符或字符串> 若两个文件在某几行有所不同，而这几行同时都包含了选项中指定的字符或字符串，则不显示这两个文件的差异。
- -i或--ignore-case 不检查大小写的不同。
- -l或--paginate 将结果交由pr程序来分页。
- -n或--rcs 将比较结果以RCS的格式来显示。
- -N或--new-file 在比较目录时，若文件A仅出现在某个目录中，预设会显示：
- Only in目录：文件A若使用-N参数，则diff会将文件A与一个空白的文件比较。
- -p 若比较的文件为C语言的程序码文件时，显示差异所在的函数名称。
- -P或--unidirectional-new-file 与-N类似，但只有当第二个目录包含了一个第一个目录所没有的文件时，才会将这个文件与空白的文件做比较。
- -q或--brief 仅显示有无差异，不显示详细的信息。
- -r或--recursive 比较子目录中的文件。

- -s或--report-identical-files　若没有发现任何差异，仍然显示信息。
- -S<文件>或--starting-file<文件>　在比较目录时，从指定的文件开始比较。
- -t或--expand-tabs　在输出时，将tab字符展开。
- -T或--initial-tab　在每行前面加上tab字符以便对齐。
- -u,-U<列数>或--unified=<列数>　以合并的方式来显示文件内容的不同。
- -v或--version　显示版本信息。
- -w或--ignore-all-space　忽略全部的空格字符。
- -W<宽度>或--width<宽度>　在使用-y参数时，指定栏宽。
- -x<文件名或目录>或--exclude<文件名或目录>　不比较选项中所指定的文件或目录。
- -X<文件>或--exclude-from<文件>　您可以将文件或目录类型存成文本文件，然后在=<文件>中指定此文本文件。
- -y或--side-by-side　以并列的方式显示文件的异同之处。
- --help　显示帮助。
- --left-column　在使用-y参数时，若两个文件某一行内容相同，则仅在左侧的栏位显示该行内容。
- --suppress-common-lines　在使用-y参数时，仅显示不同之处。

实例1：比较两个文件

```
[root@localhost test3]# diff log2014.log log2013.log
3c3
< 2014-03
---
> 2013-03
8c8
< 2013-07
---
> 2013-08
11,12d10
< 2013-11
< 2013-12
```

上面的"3c3"和"8c8"表示log2014.log和log20143log文件在3行和第8行内容有所不同；"11,12d10"表示第一个文件比第二个文件多了第11和12行。

实例2：并排格式输出

```
[root@localhost test3]# diff log2014.log log2013.log  -y -W 50
2013-01                 2013-01
2013-02                 2013-02
2014-03               | 2013-03
2013-04                 2013-04
2013-05                 2013-05
2013-06                 2013-06
2013-07                 2013-07
2013-07               | 2013-08
2013-09                 2013-09
2013-10                 2013-10
2013-11               <
2013-12               <
[root@localhost test3]# diff log2013.log log2014.log  -y -W 50
2013-01                 2013-01
2013-02                 2013-02
2013-03               | 2014-03
2013-04                 2013-04
2013-05                 2013-05
2013-06                 2013-06
2013-07                 2013-07
2013-08               | 2013-07
2013-09                 2013-09
2013-10                 2013-10
                      > 2013-11
                      > 2013-12
```

说明：

- "|"表示前后2个文件内容有不同
- "<"表示后面文件比前面文件少了1行内容
- ">"表示后面文件比前面文件多了1行内容

Linux diffstat命令

Linux diffstat命令根据diff的比较结果，显示统计数字。

diffstat读取diff的输出结果，然后统计各文件的插入，删除，修改等差异计量。

语法

```
diff [-wV][-n <文件名长度>][-p <文件名长度>]
```

参数：

- -n<文件名长度>　指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。
- -p<文件名长度>　与-n参数相同，但此处的<文件名长度>包括了文件的路径。
- -w　指定输出时栏位的宽度。
- -V　显示版本信息。

实例

用户也可以直接使用"|"将diff指令所输出的结果直接送给diffstat指令进行统计结果的显示。

使用该指令时，若所比较的文件或者子目录不在当前目录下，则应该使用其完整路径。

将目录"test1"和"test2"下的同名文件"testf.txt"使用diff指令进行比较。然后使用diffstat指令对结果进行统计显示，输入如下命令：

```
$ diff test1 test2 | diffstat   #进行比较结果的统计显示
```

注意：使用这条命令可以非常方便地实现统计显示的功能。

对于查看文件中的内容，用户可以通过指令"cat"进行查看即可，具体操作如下：

```
$ cat test1/testf.txt            #查看test1/testf的内容
abc
def
ghi
jkl
mno
pqr
stu
vws
$ cat test2/testf.txt            #查看test2/testf的内容
abc
def
ghi
jkl
mno
```

从上面的文件内容显示，可以看到两个文件内容的差别。现在来运行刚才的命令，对文件比较的结果进行统计显示，结果如下：

```
testfile | 2 +-             #统计信息输出显示
1 file changed, 1 insertion(+), 1 deletion(-)
```

Linux file命令

Linux file命令用于辨识文件类型。

通过file指令，我们得以辨识该文件的类型。

语法

```
file [-beLvz][-f <名称文件>][-m <魔法数字文件>...][文件或目录...]
```

参数：

- -b　列出辨识结果时，不显示文件名称。
- -c　详细显示指令执行过程，便于排错或分析程序执行的情形。
- -f<名称文件>　指定名称文件，其内容有一个或多个文件名称呢感，让file依序辨识这些文件，格式为每列一个文件名称。
- -L　直接显示符号连接所指向的文件的类别。
- -m<魔法数字文件>　指定魔法数字文件。
- -v　显示版本信息。
- -z　尝试去解读压缩文件的内容。
- [文件或目录...] 要确定类型的文件列表，多个文件之间使用空格分开，可以使用shell通配符匹配多个文件。

实例

显示文件类型：

```
[root@localhost ~]# file install.log
install.log: UTF-8 Unicode text

[root@localhost ~]# file -b install.log      <== 不显示文件名称
UTF-8 Unicode text

[root@localhost ~]# file -i install.log      <== 显示MIME类别。
install.log: text/plain; charset=utf-8

[root@localhost ~]# file -b -i install.log
text/plain; charset=utf-8
```

显示符号链接的文件类型

```
[root@localhost ~]# ls -l /var/mail
lrwxrwxrwx 1 root root 10 08-13 00:11 /var/mail -> spool/mail

[root@localhost ~]# file /var/mail
/var/mail: symbolic link to `spool/mail'

[root@localhost ~]# file -L /var/mail
/var/mail: directory

[root@localhost ~]# file /var/spool/mail
/var/spool/mail: directory

[root@localhost ~]# file -L /var/spool/mail
/var/spool/mail: directory
```

Linux find命令

Linux find命令用来在指定目录下查找文件。任何位于参数之前的字串都将被视为欲查找的目录名。如果使用该命令时，不设置任何参数，则find命令将在当前目录下查找子目录与文件。并且将查找到的子目录和文件全部进行显示。

语法

```
find   path   -option   [   -print ]   [ -exec   -ok   command ]   {} ;
```

参数说明 :

find 根据下列规则判断 path 和 expression，在命令列上第一个 - ( ) , ! 之前的部份为 path，之后的是 expression。如果 path 是空字串则使用目前路径，如果 expression 是空字串则使用 -print 为预设 expression。

expression 中可使用的选项有二三十个之多，在此只介绍最常用的部份。

-mount, -xdev : 只检查和指定目录在同一个文件系统下的文件，避免列出其它文件系统中的文件

-amin n : 在过去 n 分钟内被读取过

-anewer file : 比文件 file 更晚被读取过的文件

-atime n : 在过去 n 天过读取过的文件

-cmin n : 在过去 n 分钟内被修改过

-cnewer file :比文件 file 更新的文件

-ctime n : 在过去 n 天过修改过的文件

-empty : 空的文件-gid n or -group name : gid 是 n 或是 group 名称是 name

-ipath p, -path p : 路径名称符合 p 的文件，ipath 会忽略大小写

-name name, -iname name : 文件名称符合 name 的文件。iname 会忽略大小写

-size n : 文件大小 是 n 单位，b 代表 512 位元组的区块，c 表示字元数，k 表示 kilo bytes，w 是二个位元组。-type c : 文件类型是 c 的文件。

d: 目录

c: 字型装置文件

b: 区块装置文件

p: 具名贮列

f: 一般文件

l: 符号连结

s: socket

-pid n : process id 是 n 的文件

你可以使用 ( ) 将运算式分隔，并使用下列运算。

exp1 -and exp2

! expr

-not expr

exp1 -or exp2

exp1, exp2

实例

将目前目录及其子目录下所有延伸档名是 c 的文件列出来。

```
# find . -name "*.c"
```

将目前目录其其下子目录中所有一般文件列出

```
# find . -ftype f
```

将目前目录及其子目录下所有最近 20 分钟内更新过的文件列出

```
# find . -ctime -20
```

查找/var/logs目录中更改时间在7日以前的普通文件，并在删除之前询问它们：

```
$ find /var/logs -type f -mtime +7 -ok rm { } ;
```

查找前目录中文件属主具有读、写权限，并且文件所属组的用户和其他用户具有读权限的文件：

```
$ find . -type f -perm 644 -exec ls -l { } ;
```

为了查找系统中所有文件长度为0的普通文件，并列出它们的完整路径：

```
$ find / -type f -size 0 -exec ls -l { } ;
```

查找/var/logs目录中更改时间在7日以前的普通文件，并在删除之前询问它们:

```
$ find /var/logs -type f -mtime +7 -ok rm { } ;
```

Linux git命令

Linux git命令是文字模式下的文件管理员。

git是用来管理文件的程序，它十分类似DOS下的Norton Commander，具有互动式操作界面。它的操作方法和Norton Commander几乎一样。

语法

```
git
```

操作说明：

- F1 ：执行info指令，查询指令相关信息，会要求您输入欲查询的名称。
- F2 ：执行cat指令，列出文件内容。
- F3 ：执行gitview指令，观看文件内容。
- F4 ：执行vi指令，编辑文件内容。
- F5 ：执行cp指令，复制文件或目录，会要求您输入目标文件或目录。
- F6 ：执行mv指令，移动文件或目录，或是更改其名称，会要求您输入目标文件或目录。
- F7 ：执行mkdir指令，建立目录。
- F8 ：执行rm指令，删除文件或目录。
- F9 ：执行make指令，批处理执行指令或编译程序时，会要求您输入相关命令。
- F10 ：离开git文件管理员。

Linux gitview命令

Linux gitview命令用于观看文件的内容，它会同时显示十六进制和ASCII格式的字码。

语法

```
gitview [-bchilv][文件]
```

参数：

- -b 　单色模式，不使用ANSI控制码显示彩色。
- -c 　彩色模式，使用ANSI控制码显示色彩。
- -h 　在线帮助。
- -i 　显示存放gitview程序的所在位置。
- -l 　不使用先前的显示字符。
- -v 　显示版本信息。

实例

使用指令gitview以彩色模式观看文件"/home/ rootlocal/demo.txt"中的内容，输入如下命令：

```
$ gitview -c /home/rootlocal/demo.txt      #使用gitview指令观看指定文件内容
```

Linux indent命令

Linux indent命令用于调整C原始代码文件的格式。

indent可辨识C的原始代码文件，并加以格式化，以方便程序设计师阅读。

语法

```
indent [参数][源文件] 或 indent [参数][源文件][-o 目标文件]
```

参数：

- -bad或--blank-lines-after-declarations 在声明区段或加上空白行。
- -bap或--blank-lines-after-procedures 在程序或加上空白行。
- -bbb或--blank-lines-after-block-comments 在注释区段后加上空白行。
- -bc或--blank-lines-after-commas 在声明区段中，若出现逗号即换行。
- -bl或--braces-after-if-line if(或是else,for等等)与后面执行区段的"{"不同行，且"}"自成一行。
- -bli<缩排格数>或--brace-indent<缩排格数> 设置{ }缩排的格数。
- -br或--braces-on-if-line if(或是else,for等等)与后面执行跛段的"{"不同行，且"}"自成一行。
- -bs或--blank-before-sizeof 在sizeof之后空一格。
- -c<栏数>或--comment-indentation<栏数> 将注释置于程序码右侧指定的栏位。
- -cd<栏数>或--declaration-comment-column<栏数> 将注释置于声明右侧指定的栏位。
- -cdb或--comment-delimiters-on-blank-lines 注释符号自成一行。
- -ce或--cuddle-else 将else置于"}"(if执行区段的结尾)之后。
- -ci<缩排格数>或--continuation-indentation<缩排格数> 叙述过长而换行时，指定换行后缩排的格数。
- -cli<缩排格数>或--case-indentation-<缩排格数> 使用case时，switch缩排的格数。
- -cp<栏数>或-else-endif-column<栏数> 将注释置于else与elseif叙述右侧定的栏位。
- -cs或--space-after-cast 在cast之后空一格。
- -d<缩排格数>或-line-comments-indentation<缩排格数> 针对不是放在程序码右侧的注释，设置其缩排格数。
- -di<栏数>或--declaration-indentation<栏数> 将声明区段的变量置于指定的栏位。
- -fc1或--format-first-column-comments 针对放在每行最前端的注释，设置其格式。
- -fca或--format-all-comments 设置所有注释的格式。
- -gnu或--gnu-style 指定使用GNU的格式，此为预设值。
- -i<格数>或--indent-level<格数> 设置缩排的格数。
- -ip<格数>或--parameter-indentation<格数> 设置参数的缩排格数。

- -kr或--k-and-r-style　指定使用Kernighan&Ritchie的格式。
- -lp或--continue-at-parentheses　叙述过长而换行，且叙述中包含了括弧时，将括弧中的每行起始栏位内容垂直对其排列。
- -nbad或--no-blank-lines-after-declarations　在声明区段后不要加上空白行。
- -nbap或--no-blank-lines-after-procedures　在程序后不要加上空白行。
- -nbbb或--no-blank-lines-after-block-comments　在注释区段后不要加上空白行。
- -nbc或--no-blank-lines-after-commas　在声明区段中，即使出现逗号，仍旧不要换行。
- -ncdb或--no-comment-delimiters-on-blank-lines　注释符号不要自成一行。
- -nce或--dont-cuddle-else　不要将else置于"}"之后。
- -ncs或--no-space-after-casts　不要在cast之后空一格。
- -nfc1或--dont-format-first-column-comments　不要格式化放在每行最前端的注释。
- -nfca或--dont-format-comments　不要格式化任何的注释。
- -nip或--no-parameter-indentation　参数不要缩排。
- -nlp或--dont-line-up-parentheses　叙述过长而换行，且叙述中包含了括弧时，不用将括弧中的每行起始栏位垂直对其排列。
- -npcs或--no-space-after-function-call-names　在调用的函数名称之后，不要加上空格。
- -npro或--ignore-profile　不要读取indent的配置文件.indent.pro。
- -npsl或--dont-break-procedure-type　程序类型与程序名称放在同一行。
- -nsc或--dont-star-comments　注解左侧不要加上星号(*)。
- -nsob或--leave-optional-semicolon　不用处理多余的空白行。
- -nss或--dont-space-special-semicolon　若for或while区段仅有一行时，在分号前不加上空格。
- -nv或--no-verbosity　不显示详细的信息。
- -orig或--original　使用Berkeley的格式。
- -pcs或--space-after-procedure-calls　在调用的函数名称与"{"之间加上空格。
- -psl或--procnames-start-lines　程序类型置于程序名称的前一行。
- -sc或--start-left-side-of-comments　在每行注释左侧加上星号(*)。
- -sob或--swallow-optional-blank-lines　删除多余的空白行。
- -ss或--space-special-semicolon　若for或swile区段今有一行时，在分号前加上空格。
- -st或--standard-output　将结果显示在标准输出设备。
- -T　数据类型名称缩排。
- -ts<格数>或--tab-size<格数>　设置tab的长度。
- -v或--verbose　执行时显示详细的信息。
- -version　显示版本信息。

Indent代码格式化说明

使用的**indent**参数	值	含义
--blank-lines-after-declarations	bad	变量声明后加空行

--blank-lines-after-procedures	bap	函数结束后加空行
--blank-lines-before-block-comments	bbb	块注释前加空行
--break-before-boolean-operator	bbo	较长的行，在逻辑运算符前分行
--blank-lines-after-commas	nbc	变量声明中，逗号分隔的变量不分行
--braces-after-if-line	bl	"if"和"{"分做两行
--brace-indent 0	bli0	"{"不继续缩进
--braces-after-struct-decl-line	bls	定义结构，"struct"和"{"分行
--comment-indentationn	c33	语句后注释开始于行33
--declaration-comment-columnn	cd33	变量声明后注释开始于行33
--comment-delimiters-on-blank-lines	ncdb	不将单行注释变为块注释
--cuddle-do-while	ncdw	"do --- while"的"while"和其前面的"}"另起一行
--cuddle-else	nce	"else"和其前面的"}"另起一行
--case-indentation 0	cli0	switch中的case语句所进0个空格
--else-endif-columnn	cp33	#else, #endif后面的注释开始于行33
--space-after-cast	cs	在类型转换后面加空格
--line-comments-indentation n	d0	单行注释（不从1列开始的），不向左缩进
--break-function-decl-args	nbfda	关闭：函数的参数一个一行
--declaration-indentationn	di2	变量声明，变量开始于2行，即不必对齐
--format-first-column-comments	nfc1	不格式化起于第一行的注释
--format-all-comments	nfca	不开启全部格式化注释的开关
--honour-newlines	hnl	Prefer to break long lines at the position of newlines in the input.
--indent-leveln	i4	设置缩进多少字符，如果为tab的整数倍，用tab来缩进，否则用空格填充。
--parameter-indentationn	ip5	旧风格的函数定义中参数说明缩进5个空格
--line-length 75	l75	非注释行最长75

--continue-at-parentheses	lp	续行从上一行出现的括号开始
--space-after-procedure-calls	pcs	函数和"("之间插入一个空格
--space-after-parentheses	nprs	在"（"后"）"前不插入空格
--procnames-start-lines	psl	将函数名和返回类型放在两行定义
--space-after-for	saf	for后面有空格
--space-after-if	sai	if后面有空格
--space-after-while	saw	while后面有空格
--start-left-side-of-comments	nsc	不在生成的块注释中加*
--swallow-optional-blank-lines	nsob	不去掉可添加的空行
--space-special-semicolon	nss	一行的for或while语句，在";"前不加空。
--tab-size	ts4	一个tab为4个空格（要能整除"-in"）
--use-tabs	ut	使用tab来缩进

Linux cut命令

Linux cut命令用于显示每行从开头算起 num1 到 num2 的文字。

语法

```
cut  [-bn] [file]
cut [-c] [file]
cut [-df] [file]
```

使用说明:

cut 命令从文件的每一行剪切字节、字符和字段并将这些字节、字符和字段写至标准输出。

如果不指定 File 参数，cut 命令将读取标准输入。必须指定 -b、-c 或 -f 标志之一。

参数:

- -b ：以字节为单位进行分割。这些字节位置将忽略多字节字符边界，除非也指定了 -n 标志。
- -c ：以字符为单位进行分割。
- -d ：自定义分隔符，默认为制表符。
- -f ：与-d一起使用，指定显示哪个区域。
- -n ：取消分割多字节字符。仅和 -b 标志一起使用。如果字符的最后一个字节落在由 -b 标志的 List 参数指示的
  范围之内，该字符将被写出；否则，该字符将被排除

实例

当你执行who命令时，会输出类似如下的内容：

```
$ who
rocrocket :0           2009-01-08 11:07
rocrocket pts/0        2009-01-08 11:23 (:0.0)
rocrocket pts/1        2009-01-08 14:15 (:0.0)
```

如果我们想提取每一行的第3个字节，就这样：

```
$ who|cut -b 3
c
c
```

Linux ln命令

Linux ln命令是一个非常重要命令，它的功能是为某一个文件在另外一个位置建立一个同步的链接。

当我们需要在不同的目录，用到相同的文件时，我们不需要在每一个需要的目录下都放一个必须相同的文件，我们只要在某个固定的目录，放上该文件，然后在 其它的目录下用ln命令链接（link）它就可以，不必重复的占用磁盘空间。

语法

```
ln [参数][源文件或目录][目标文件或目录]
```

其中参数的格式为

[-bdfinsvF] [-S backup-suffix] [-V {numbered,existing,simple}]

[--help] [--version] [--]

命令功能：
Linux文件系统中，有所谓的链接(link)，我们可以将其视为档案的别名，而链接又可分为两种 : 硬链接(hard link)与软链接(symbolic link)，硬链接的意思是一个档案可以有多个名称，而软链接的方式则是产生一个特殊的档案，该档案的内容是指向另一个档案的位置。硬链接是存在同一个文件系统中，而软链接却可以跨越不同的文件系统。

不论是硬链接或软链接都不会将原本的档案复制一份，只会占用非常少量的磁碟空间。

软链接：

- 1.软链接，以路径的形式存在。类似于Windows操作系统中的快捷方式
- 2.软链接可以 跨文件系统 ，硬链接不可以
- 3.软链接可以对一个不存在的文件名进行链接
- 4.软链接可以对目录进行链接

硬链接：

- 1.硬链接，以文件副本的形式存在。但不占用实际空间。
- 2.不允许给目录创建硬链接
- 3.硬链接只有在同一个文件系统中才能创建

命令参数

必要参数：

- -b 删除，覆盖以前建立的链接
- -d 允许超级用户制作目录的硬链接
- -f 强制执行
- -i 交互模式，文件存在则提示用户是否覆盖
- -n 把符号链接视为一般目录
- -s 软链接(符号链接)
- -v 显示详细的处理过程

选择参数：

- -S "-S<字尾备份字符串> "或 "--suffix=<字尾备份字符串>"
- -V "-V<备份方式>"或"--version-control=<备份方式>"
- --help 显示帮助信息
- --version 显示版本信息

实例

给文件创建软链接，为log2013.log文件创建软链接link2013，如果log2013.log丢失，link2013将失效：

```
ln -s log2013.log link2013
```

输出：

```
[root@localhost test]# ll
-rw-r--r-- 1 root bin      61 11-13 06:03 log2013.log
[root@localhost test]# ln -s log2013.log link2013
[root@localhost test]# ll
lrwxrwxrwx 1 root root     11 12-07 16:01 link2013 -> log2013.log
-rw-r--r-- 1 root bin      61 11-13 06:03 log2013.log
```

给文件创建硬链接，为log2013.log创建硬链接ln2013，log2013.log与ln2013的各项属性相同

```
ln log2013.log ln2013
```

输出：

```
[root@localhost test]# ll
lrwxrwxrwx 1 root root     11 12-07 16:01 link2013 -> log2013.log
-rw-r--r-- 1 root bin      61 11-13 06:03 log2013.log
[root@localhost test]# ln log2013.log ln2013
[root@localhost test]# ll
lrwxrwxrwx 1 root root     11 12-07 16:01 link2013 -> log2013.log
-rw-r--r-- 2 root bin      61 11-13 06:03 ln2013
-rw-r--r-- 2 root bin      61 11-13 06:03 log2013.log
```

Linux less命令

less 与 more 类似，但使用 less 可以随意浏览文件，而 more 仅能向前移动，却不能向后移动，而且 less 在查看之前不会加载整个文件。

语法

```
less [参数] 文件
```

参数说明：

- -b <缓冲区大小> 设置缓冲区的大小
- -e 当文件显示结束后，自动离开
- -f 强迫打开特殊文件，例如外围设备代号、目录和二进制文件
- -g 只标志最后搜索的关键词
- -i 忽略搜索时的大小写
- -m 显示类似more命令的百分比
- -N 显示每行的行号
- -o <文件名> 将less 输出的内容在指定文件中保存起来
- -Q 不使用警告音
- -s 显示连续空行为一行
- -S 行过长时间将超出部分舍弃
- -x <数字> 将"tab"键显示为规定的数字空格
- /字符串：向下搜索"字符串"的功能
- ?字符串：向上搜索"字符串"的功能
- n：重复前一个搜索（与 / 或 ? 有关）
- N：反向重复前一个搜索（与 / 或 ? 有关）
- b 向后翻一页
- d 向后翻半页
- h 显示帮助界面
- Q 退出less 命令
- u 向前滚动半页
- y 向前滚动一行
- 空格键 滚动一行
- 回车键 滚动一页
- [pagedown]： 向下翻动一页
- [pageup]： 向上翻动一页

实例

1、查看文件

```
less log2013.log
```

2、ps查看进程信息并通过less分页显示

```
ps -ef |less
```

3、查看命令历史使用记录并通过less分页显示

```
[root@localhost test]# history | less
22  scp -r tomcat6.0.32 root@192.168.120.203:/opt/soft
23  cd ..
24  scp -r web root@192.168.120.203:/opt/
25  cd soft
26  ls
......省略......
```

4、浏览多个文件

```
less log2013.log log2014.log
```

说明：
输入 ：n后，切换到 log2014.log
输入 ：p 后，切换到log2013.log

附加备注

1.全屏导航

- ctrl + F - 向前移动一屏
- ctrl + B - 向后移动一屏
- ctrl + D - 向前移动半屏
- ctrl + U - 向后移动半屏

2.单行导航

- j - 向前移动一行
- k - 向后移动一行

3.其它导航

- G - 移动到最后一行
- g - 移动到第一行

- q / ZZ - 退出 less 命令

4.其它有用的命令

- v - 使用配置的编辑器编辑当前文件
- h - 显示 less 的帮助文档
- &pattern - 仅显示匹配模式的行，而不是整个文件

5.标记导航

当使用 less 查看大文件时，可以在任何一个位置作标记，可以通过命令导航到标有特定标记的文本位置：

- ma - 使用 a 标记文本的当前位置
- 'a - 导航到标记 a 处

Linux locate命令

Linux locate命令用于查找符合条件的文档，他会去保存文档和目录名称的数据库内，查找合乎范本样式条件的文档或目录。

一般情况我们只需要输入 **locate your_file_name** 即可查找指定文件。

语法

```
locate [-d ][--help][--version][范本样式...]
```

参数：

- -d或--database= 配置locate指令使用的数据库。locate指令预设的数据库位于/var/lib/slocate目录里，文档名为slocate.db，您可使用 这个参数另行指定。
- --help 在线帮助。
- --version 显示版本信息。

实例

查找passwd文件，输入以下命令：

```
locate passwd
```

附加说明

locate与find 不同: find 是去硬盘找，locate 只在/var/lib/slocate资料库中找。

locate的速度比find快，它并不是真的查找，而是查数据库，一般文件数据库在/var/lib/slocate/slocate.db中，所以locate的查找并不是实时的，而是以数据库的更新为准，一般是系统自己维护，也可以手工升级数据库 ，命令为：

```
locate -u
```

Linux lsattr命令

Linux lsattr命令用于显示文件属性。

用chattr执行改变文件或目录的属性，可执行lsattr指令查询其属性。

语法

```
lsattr [-adlRvV][文件或目录...]
```

参数：

- -a 　显示所有文件和目录，包括以"."为名称开头字符的额外内建，现行目录"."与上层目录".."。
- -d 　显示，目录名称，而非其内容。
- -l 　此参数目前没有任何作用。
- -R 　递归处理，将指定目录下的所有文件及子目录一并处理。
- -v 　显示文件或目录版本。
- -V 　显示版本信息。

实例

1、用chattr命令防止系统中某个关键文件被修改：

```
# chattr +i /etc/resolv.conf
```

然后用mv /etc/resolv.conf等命令操作于该文件，都是得到Operation not permitted 的结果。

vim编辑该文件时会提示W10: Warning: Changing a readonly file错误。要想修改此文件就要把i属性去掉：

```
chattr -i /etc/resolv.conf
```

使用 lsattr 命令来显示文件属性：

```
# lsattr /etc/resolv.conf
```

输出结果为:

```
----i-------- /etc/resolv.conf
```

2、让某个文件只能往里面追加数据，但不能删除，适用于各种日志文件：

```
# chattr +a /var/log/messages
```

Linux mattrib命令

Linux mattrib命令用来变更或显示MS-DOS文件的属性。

mattrib为mtools工具指令，模拟MS-DOS的attrib指令，可变更MS-DOS文件的属性。

语法

```
mattrib [-a|+a] [-h|+h] [-r|+r] [-s|+s] [-/] [-X] msdosfile [ msdosfiles ... ]
```

参数：

- -a/+a 除去/设定备份属性。
- -h/+h 除去/设定隐藏属性。
- -r/+r 除去/设定唯读属性。
- -s/+s 除去/设定系统属性。
- -/ 递回的处理包含所有子目录下的档案。
- -X 以较短的格式输出结果。

实例

列出 A 槽 MSDOS 格式磁片上所有文件的属性。

```
mattrib a:
```

除去 A 槽磁片上 msdos.sys 档案的隐藏、系统与唯读属性。

```
mattrib -h -s -r a:msdos.sys
```

除去 A 槽磁片上包含子目录下所有档案的唯读属性。

```
mattrib -r -/ a:*.*
```

Linux mc命令

Linux mc命令用于提供一个菜单式的文件管理程序。

执行mc之后，将会看到菜单式的文件管理程序，共分成4 个部分。

语法

```
mc [-abcdfhkPstuUVx][-C <参数>][-l <文件>][-v <文件>][目录]
```

参　数：

- -a 　当mc程序画线时不用绘图字符画线。
- -b 　使用单色模式显示。
- -c 　使用彩色模式显示。
- -C<参数> 　指定显示的颜色。
- -d 　不使用鼠标。
- -f 　显示mc函数库所在的目录。
- -h 　显示帮助。
- -k 　重设softkeys成预设置。
- -l<文件> 　在指定文件中保存ftpfs对话窗的内容。
- -P 　程序结束时，列出最后的工作目录。
- -s 　用慢速的终端机模式显示，在这模式下将减少大量的绘图及文字显示。
- -t 　使用TEMPCAP变量设置终端机，而不使用预设置。
- -u 　不用目前的shell程序。
- -U 　使用目前的shell程序。
- -v<文件> 　使用mc的内部编辑器来显示指定的文件。
- -V 　显示版本信息。
- -x 　指定以xterm模式显示。

Linux MC 相关操作

命令按键	描 述
F9 or Esc+9	激活菜单栏
Tab	在两个窗口间移动
F10 or Esc+0	退出MC
Control-Enter or Alt-Enter	可以将文件名拷贝到命令行
F1 or Esc+1	打开帮助页面

虽然MC很好用，不过我还是建议大家使用命令行工具！

Linux mdel命令

Linux mdel命令用来删除 MSDOS 格式的档案。

在删除只读之前会有提示信息产生。

语法

```
mdel [-v] msdosfile [ msdosfiles ... ]
```

参数：

- -v 显示更多的讯息。

实例

将 A 槽磁片根目录中的 autoexec.bat 删除。

```
mdel a:autoexec.bat .
```

Linux mdir命令

Linux mdir命令用于显示MS-DOS目录。

mdir为mtools工具指令，模拟MS-DOS的dir指令，可显示MS-DOS文件系统中的目录内容。

语法

```
mdir [-afwx/][目录]
```

参数：

- -/ 显示目录下所有子目录与文件。
- -a 显示隐藏文件。
- -f 不显示磁盘所剩余的可用空间。
- -w 仅显示目录或文件名称，并以横排方式呈现，以便一次能显示较多的目录或文件。
- -X 仅显示目录下所有子目录与文件的完整路径，不显示其他信息。

实例

显示a盘中的内容

```
$ mdir -/ a:\*
```

以上命令执行后，mdir将显示指定盘"a:\"中的所有子目录及其中的文件信息，如下所示：

```
Volume in drive A has no label  #加载信息
Volume Serial Number is 13D2~055C
Directory for A:\                   #以下为目录信息
./TEST <DIR> 2011-08-23 16:59
#显示格式为文件名，目录大小，修改时间
AUTORUN.INF 265 2011-08-23 16:53
AUTORUN.BAT 43 2011-08-23 16:56
3 files 308 bytes               #统计总大小
724 325 bytes free              #剩余空间
```

Linux mktemp命令

Linux mktemp命令用于建立暂存文件。

mktemp建立的一个暂存文件，供shell script使用。

语法

```
mktemp [-qu][文件名参数]
```

参数：

- -q 执行时若发生错误，不会显示任何信息。
- -u 暂存文件会在mktemp结束前先行删除。
- [文件名参数] 文件名参数必须是以"自订名称.XXXXXX"的格式。

实例

使用mktemp 命令生成临时文件时，文件名参数应当以"文件名.XXXX"的形式给出，mktemp 会根据文件名参数建立一个临时文件。在命令行提示符输入如下命令：

```
mktemp tmp.xxxx #生成临时文件
```

使用该命令后，可使用dir 或ls看当前目录，得到如下结果：

```
cmd@cmd-desktop:~$ mktemp tmp.xxxx #生成临时文件
cmd@cmd-desktop:~$dir #查看当前目录
file test testfile testfile1 tmp.3847 #生成了tmp.3847
```

由此可见，生成的临时文件为tmp.3847，其中，文件名参数中的"XXXX"被4 个随机产生的字符所取代。

Linux more命令

Linux more 命令类似 cat ，不过会以一页一页的形式显示，更方便使用者逐页阅读，而最基本的指令就是按空白键（space）就往下一页显示，按 b 键就会往回（back）一页显示，而且还有搜寻字串的功能（与 vi 相似），使用中的说明文件，请按 h 。

语法

```
more [-dlfpcsu] [-num] [+/pattern] [+linenum] [fileNames..]
```

参数：

- -num 一次显示的行数
- -d 提示使用者，在画面下方显示 [Press space to continue, 'q' to quit.] ，如果使用者按错键，则会显示 [Press 'h' for instructions.] 而不是 '哔' 声
- -l 取消遇见特殊字元 ^L（送纸字元）时会暂停的功能
- -f 计算行数时，以实际上的行数，而非自动换行过后的行数（有些单行字数太长的会被扩展为两行或两行以上）
- -p 不以卷动的方式显示每一页，而是先清除萤幕后再显示内容
- -c 跟 -p 相似，不同的是先显示内容再清除其他旧资料
- -s 当遇到有连续两行以上的空白行，就代换为一行的空白行
- -u 不显示下引号 （根据环境变数 TERM 指定的 terminal 而有所不同）
- +/pattern 在每个文档显示前搜寻该字串（pattern），然后从该字串之后开始显示
- +num 从第 num 行开始显示
- fileNames 欲显示内容的文档，可为复数个数

实例

逐页显示 testfile 文档内容，如有连续两行以上空白行则以一行空白行显示。

```
more -s testfile
```

从第 20 行开始显示 testfile 之文档内容。

```
more +20 testfile
```

常用操作命令

- Enter 向下n行，需要定义。默认为1行

- Ctrl+F 向下滚动一屏
- 空格键 向下滚动一屏
- Ctrl+B 返回上一屏
- = 输出当前行的行号
- ：f 输出文件名和当前行的行号
- V 调用vi编辑器
- !命令 调用Shell，并执行命令
- q 退出more

Linux mmove命令

Linux mmove命令用于在MS-DOS文件系统中，移动文件或目录，或更改名称。

mmove为mtools工具命令，模拟MS-DOS的move命令，可在MS-DOS文件系统中移动现有的文件或目录，或是更改现有文件或目录的名称。

语法

```
mmove [源文件或目录...][目标文件或目录]
```

参数说明:

- [源文件或目录...]: 执行操作的源文件或目录路径
- [目标文件或目录]: 执行操作后的目标文件或目录路径

实例

使用指令mmove将文件"autorun.bat"移动到目录"test"中，输入如下命令：

```
$ mmove autorun.bat test        #移动文件到目录test中
```

以上命令执行以后，指令mmove会将文件"autorun.bat"移动到指定目录"test"中。

注意：用户可以使用mdir指令查看移动后的文件或目录信息。

Linux mread命令

Linux mread命令用于将MS-DOS文件复制到Linux/Unix的目录中。

mread为mtools工具命令，可将MS-DOS文件复制到Linux的文件系统中。这个命令目前已经不常用，一般都使用mcopy命令来代替。

语法

```
mread [MS-DOS文件...][Linux文件或目录]
```

参数说明:

- [MS-DOS文件...]: 执行操作的DOS源文件或目录路径
- [Linux文件或目录]: 执行操作后的Linux目标文件或目录路径

实例

使用指令mread将盘"a:\"中的所有内容复制到当前工作目录下，输入如下命令：

```
$ mread a:\* ./     #将a盘上的所有文件复制到当前工作目录
```

执行该命令前，可以先使用mdir命令查看原来的目录结构。执行mread之后，可使用ls命令再次查看复制之后的文件结构，结果如下所示：

```
$ mdir -/ a:\*      #查看a盘中的文件
Volume in drive A has no label          #加载信息
Volume Serial Number is 13D2~055C
Directory for A:/                    #以下为目录信息
./TEST <DIR> 2011-08-23 16:59
#显示格式为文件名，目录大小，修改时间
AUTORUN.INF 265 2011-08-23 16:53
AUTORUN.BAT 43 2011-08-23 16:56
3 files 308 bytes                       #统计总大小
724 325 bytes free                   #剩余空间
$ mread A:\* ./     #将a盘上所有文件复制到当前工作目录
$ ls                                    #查看文件或子目录信息
TEST AUTORUN.INF AUTORUN.BAT            #显示复制后的内容
```

Linux mren命令

Linux mren命令用于更改MS-DOS文件或目录的名称，或是移动文件或目录。

mren为MS-DOS工具指令，与DOS下的ren指令相似，可以实现更改MS-DOS文件或目录名称。

源文件必须是磁盘上已经存在的文件，若忽略盘符及路径，则表示当前盘及当前目录的文件。

新文件名是所要更换的文件名称。新文件名称前不可以加与源文件不同的盘符及路径，因为该命令只能更改同一盘上的文件名称。

语法

```
mren [源文件或目录...][目标文件或目录]
```

参数说明：

- [源文件或目录...]：执行操作的源文件名或者源文件路径

实例

使用指令mren将a盘下的文件"autorun.bat"的文件名修改为"auto.bat"，输入如下命令：

```
$ mren a:\autorun.bat auto.bat
#将文件autorun.bat重命名为auto.bat
```

使用该命令前后使用mdir命令查看并对比，得到结果如下：

```
$ mdir -/ a:\*                  #查看a盘中的文件
Volume in drive A has no label  #加载信息
Volume Serial Number is 13D2~055C
Directory for A:\                   #以下为目录信息
./TEST <DIR> 2011-08-23 16:59       #文件名，目录大小，修改时间
AUTORUN.BAT 43 2011-08-23 16:56
3 files 308 bytes                   #统计总大小
724 325 bytes free              #剩余空间
#将文件autorun.bat重命名为auto.bat
$ mren a:\autorun.bat auto.bat
$ mdir -/ a:\*                  #再次查看a盘中文件
Volume in drive A has no label      #加载信息
Volume Serial Number is 13D2~055C
Directory for A:\                   #以下为目录信息
./TEST <DIR> 2011-08-23 16:59       #文件名目录大小 修改时间
#文件名被改为auto.bat，修改时间改为当前系统时间
AUTO.BAT 43 2011-08-23 16:56
3 files 308 bytes                   #统计总大小
724 325 bytes free              #剩余空间
```

Linux mtools命令

Linux mtools命令用于显示mtools支持的指令。

mtools为MS-DOS文件系统的工具程序，可模拟许多MS-DOS的指令。这些指令都是mtools的符号连接，因此会有一些共同的特性。

语法

```
mtools
```

参数说明：

- -a 长文件名重复时自动更改目标文件的长文件名。
- -A 短文件名重复但长文件名不同时自动更改目标文件的短文件名。
- -o 长文件名重复时，将目标文件覆盖现有的文件。
- -O 短文件名重复但长文件名不同时，将目标文件覆盖现有的文件。
- -r 长文件名重复时，要求用户更改目标文件的长文件名。
- -R 短文件名重复但长文件名不同时，要求用户更改目标文件的短文件名。
- -s 长文件名重复时，则不处理该目标文件。
- -S 短文件名重复但长文件名不同时，则不处理该目标文件。
- -v 执行时显示详细的说明。
- -V 显示版本信息。

实例

显示 mtools软件包所支持的MS-DOS命令。

在命令提示符中直接输入mtools，可显示其所支持的MS-DOS命令，如下所示：

```
$ mtools #显示所支持的MS-DOS命令
Supported commands: #命令列表
mattrib, mbadblocks, mcat, mcd, mclasserase, mcopy, mdel, mdeltree
mdir, mdoctorfat, mdu, mformat, minfo, mlabel, mmd, mmount
mpartition, mrd, mread, mmove, mren, mshowfat, mtoolstest, mtype
mwrite, mzip
```

Linux mtoolstest命令

Linux mtoolstest命令用于测试并显示mtools的相关设置。

mtoolstest为mtools工具指令，可读取与分析mtools的配置文件，并在屏幕上显示结果。

语法

```
mtoolstest
```

实例

在命令行中直接输入mtoolstest，即可显示mtools软件包当前的配置信息，结果如下：

```
$ mtoolstest #显示mtools 软件包当前的配置信息
drive J: #mtools软件包当前的配置信息列表
#fn=0 mode=0 builtin
file="/dev/sdb4" fat_bits=16
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
mformat_only
drive Z:
#fn=0 mode=0 builtin
file="/dev/sdb4" fat_bits=16
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
mformat_only
drive X:
#fn=0 mode=0 builtin
file="$DISPLAY" fat_bits=0
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
drive A:
#fn=2 mode=128 defined in /etc/mtools.conf
file="/dev/fd0" fat_bits=0
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
exclusive
drive B:
#fn=2 mode=128 defined in /etc/mtools.conf
file="/dev/fd1" fat_bits=0
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
exclusive
drive M:
#fn=2 mode=0 defined in /etc/mtools.conf
file="/var/lib/dosemu/hdimage.first" fat_bits=0
tracks=0 heads=0 sectors=0 hidden=0
offset=0x80
partition=1
drive N:
#fn=2 mode=0 defined in /etc/mtools.conf
file="/var/lib/dosemu/fdimage" fat_bits=0
tracks=0 heads=0 sectors=0 hidden=0
offset=0x0
partition=0
mtools_fat_compatibility=0
mtools_skip_check=0
mtools_lower_case=0
```

Linux mv命令

Linux mv命令用来为文件或目录改名、或将文件或目录移入其它位置。

语法

```
mv [options] source dest
mv [options] source... directory
```

参数说明：

- -i: 若指定目录已有同名文件，则先询问是否覆盖旧文件;
- -f: 在mv操作要覆盖某已有的目标文件时不给任何指示;

mv参数设置与运行结果

命令格式	运行结果
mv 文件名 文件名	将源文件名改为目标文件名
mv 文件名 目录名	将文件移动到目标目录
mv 目录名 目录名	目标目录已存在，将源目录 移动到目标目录；目标 目录不存在则改名
mv 目录名 文件名	出错

实例

将文件 aaa 更名为 bbb :

```
mv aaa bbb
```

将info目录放入logs目录中。注意，如果logs目录不存在，则该命令将info改名为logs。

```
mv info/ logs
```

再如将/usr/student下的所有文件和目录移到当前目录下，命令行为：

```
$ mv /usr/student/*  .
```

Linux od命令

Linux od命令用于输出文件内容。

od指令会读取所给予的文件的内容，并将其内容以八进制字码呈现出来。

语法

```
od [-abcdfhilovx][-A <字码基数>][-j <字符数目>][-N <字符数目>][-s <字符串字符数>][-t <输出格式>]
```

参数：

- -a 　此参数的效果和同时指定"-ta"参数相同。
- -A<字码基数> 　选择要以何种基数计算字码。
- -b 　此参数的效果和同时指定"-toC"参数相同。
- -c 　此参数的效果和同时指定"-tC"参数相同。
- -d 　此参数的效果和同时指定"-tu2"参数相同。
- -f 　此参数的效果和同时指定"-tfF"参数相同。
- -h 　此参数的效果和同时指定"-tx2"参数相同。
- -i 　此参数的效果和同时指定"-td2"参数相同。
- -j<字符数目>或--skip-bytes=<字符数目> 　略过设置的字符数目。
- -l 　此参数的效果和同时指定"-td4"参数相同。
- -N<字符数目>或--read-bytes=<字符数目> 　到设置的字符数目为止。
- -o 　此参数的效果和同时指定"-to2"参数相同。
- -s<字符串字符数>或--strings=<字符串字符数> 　只显示符合指定的字符数目的字符串。
- -t<输出格式>或--format=<输出格式> 　设置输出格式。
- -v或--output-duplicates 　输出时不省略重复的数据。
- -w<每列字符数>或--width=<每列字符数> 　设置每列的最大字符数。
- -x 　此参数的效果和同时指定"-h"参数相同。
- --help 　在线帮助。
- --version 　显示版本信息。

实例

创建 tmp 文件：

```
$ echo abcdef g > tmp
$ cat tmp
abcdef g
```

使用 od 命令：

```
$ od -b tmp
0000000 141 142 143 144 145 146 040 147 012
0000011
```

使用单字节八进制解释进行输出，注意左侧的默认地址格式为八字节:

```
$ od -c tmp
0000000   a   b   c   d   e   f       g  \n
0000011
```

使用ASCII码进行输出，注意其中包括转义字符

```
$ od -t d1 tmp
0000000   97   98   99  100  101  102   32  103   10
0000011
```

使用单字节十进制进行解释

```
$ od -A d -c tmp
0000000   a   b   c   d   e   f       g  \n
0000009
```

Linux paste命令

Linux paste命令用于合并文件的列。

paste指令会把每个文件以列对列的方式，一列列地加以合并。

语法

```
paste [-s][-d <间隔字符>][--help][--version][文件...]
```

参数：

- -d<间隔字符>或--delimiters=<间隔字符>　用指定的间隔字符取代跳格字符。
- -s或--serial　串列进行而非平行处理。
- --help　在线帮助。
- --version　显示帮助信息。
- [文件…] 指定操作的文件路径

实例

使用paste指令将文件"file"、"testfile"、"testfile1"进行合并，输入如下命令：

```
paste file testfile testfile1 #合并指定文件的内容
```

但是，在执行以上命令之前，首先使用"cat"指令对3个文件内容进行查看，显示如下所示：

```
$ cat file                  #file文件的内容
xiongdan 200
lihaihui 233
lymlrl 231
$ cat testfile              #testfile文件的内容
liangyuanm  ss
$ cat testfile1             #testfile1文件的内容
huanggai 56
zhixi 73
```

当合并指令"$ paste file testfile testfile1"执行后，程序界面中将显示合并后的文件内容，如下所示：

```
xiongdan 200
lihaihui 233
lymlrl 231
liangyuanm  ss
huanggai 56
zhixi 73
```

若使用paste指令的参数"-s"，则可以将一个文件中的多行数据合并为一行进行显示。例如，将文件"file"中的3行数据合并为一行数据进行显示，输入如下命令

```
$ paste -s file              #合并指定文件的多行数据
```

上面的命令执行后，显示的数据内容如下所示：

```
xiongdan 200 lihaihui 233 lymlrl 231
```

注意：参数"-s"只是将testfile文件的内容调整显示方式，并不会改变原文件的内容格式。

Linux patch命令

Linux patch命令用于修补文件。

patch指令让用户利用设置修补文件的方式，修改，更新原始文件。倘若一次仅修改一个文件，可直接在指令列中下达指令依序执行。如果配合修补文件的方式则能一次修补大批文件，这也是Linux系统核心的升级方法之一。

语法

```
patch [-bceEflnNRstTuvZ][-B <备份字首字符串>][-d <工作目录>][-D <标示符号>][-F <监别列数>][-g <
```

参数：

- -b或--backup 备份每一个原始文件。
- -B<备份字首字符串>或--prefix=<备份字首字符串> 设置文件备份时，附加在文件名称前面的字首字符串，该字符串可以是路径名称。
- -c或--context 把修补数据解译成关联性的差异。
- -d<工作目录>或--directory=<工作目录> 设置工作目录。
- -D<标示符号>或--ifdef=<标示符号> 用指定的符号把改变的地方标示出来。
- -e或--ed 把修补数据解译成ed指令可用的叙述文件。
- -E或--remove-empty-files 若修补过后输出的文件其内容是一片空白，则移除该文件。
- -f或--force 此参数的效果和指定"-t"参数类似，但会假设修补数据的版本为新 版本。
- -F<监别列数>或--fuzz<监别列数> 设置监别列数的最大值。
- -g<控制数值>或--get=<控制数值> 设置以RSC或SCCS控制修补作业。
- -i<修补文件>或--input=<修补文件> 读取指定的修补问家你。
- -l或--ignore-whitespace 忽略修补数据与输入数据的跳格，空格字符。
- -n或--normal 把修补数据解译成一般性的差异。
- -N或--forward 忽略修补的数据较原始文件的版本更旧，或该版本的修补数据已使 用过。
- -o<输出文件>或--output=<输出文件> 设置输出文件的名称，修补过的文件会以该名称存放。
- -p<剥离层级>或--strip=<剥离层级> 设置欲剥离几层路径名称。
- -f<拒绝文件>或--reject-file=<拒绝文件> 设置保存拒绝修补相关信息的文件名称，预设的文件名称为.rej。
- -R或--reverse 假设修补数据是由新旧文件交换位置而产生。
- -s或--quiet或--silent 不显示指令执行过程，除非发生错误。
- -t或--batch 自动略过错误，不询问任何问题。

- -T或--set-time　此参数的效果和指定"-Z"参数类似，但以本地时间为主。
- -u或--unified　把修补数据解译成一致化的差异。
- -v或--version　显示版本信息。
- -V<备份方式>或--version-control=<备份方式>　用"-b"参数备份目标文件后，备份文件的字尾会被加上一个备份字符串，这个字符串不仅可用"-z"参数变更，当使用"-V"参数指定不同备份方式时，也会产生不同字尾的备份字符串。
- -Y<备份字首字符串>或--basename-prefix=--<备份字首字符串>　设置文件备份时，附加在文件基本名称开头的字首字符串。
- -z<备份字尾字符串>或--suffix=<备份字尾字符串>　此参数的效果和指定"-B"参数类似，差别在于修补作业使用的路径与文件名若为src/linux/fs/super.c，加上"backup/"字符串后，文件super.c会备份于/src/linux/fs/backup目录里。
- -Z或--set-utc　把修补过的文件更改，存取时间设为UTC。
- --backup-if-mismatch　在修补数据不完全吻合，且没有刻意指定要备份文件时，才备份文件。
- --binary　以二进制模式读写数据，而不通过标准输出设备。
- --help　在线帮助。
- --nobackup-if-mismatch　在修补数据不完全吻合，且没有刻意指定要备份文件时，不要备份文件。
- --verbose　详细显示指令的执行过程。

实例

使用patch指令将文件"testfile1"升级，其升级补丁文件为"testfile.patch"，输入如下命令：

```
$ patch -p0 testfile1 testfile.patch    #使用补丁程序升级文件
```

使用该命令前，可以先使用指令"cat"查看"testfile1"的内容。在需要修改升级的文件与原文件之间使用指令"diff"比较可以生成补丁文件。具体操作如下所示：

```
$ cat testfile1                 #查看testfile1的内容
Hello,This is the firstfile!
$ cat testfile2                 #查看testfile2的内容
Hello,Thisisthesecondfile!
$ diff testfile1 testfile2          #比较两个文件
1c1
<Hello,Thisisthefirstfile!
---
>Hello,Thisisthesecondfile!
#将比较结果保存到tetsfile.patch文件
$ diff testfile1 testfile2>testfile.patch
$ cat testfile.patch                #查看补丁包的内容
1c1
<Hello,Thisisthefirstfile!
---
>Hello,Thisisthesecondfile!
#使用补丁包升级testfile1文件
$ patch -p0 testfile1 testfile.patch
patching file testfile1
$cat testfile1                  #再次查看testfile1的内容
#testfile1文件被修改为与testfile2一样的内容
Hello,This is the secondfile!
```

注意：上述命令代码中，"$ diff testfile1 testfile2>testfile. patch"所使用的操作符"＞"表示将该操作符左边的文件数据写入到右边所指向的文件中。在这里，即是指将两个文件比较后的结果写入到文件"testfile.patch"中。

Linux rcp命令

Linux rcp命令用于复制远程文件或目录。

rcp指令用在远端复制文件或目录，如同时指定两个以上的文件或目录，且最后的目的地是一个已经存在的目录，则它灰把前面指定的所有文件或目录复制到该目录中。

语法

```
rcp [-pr][源文件或目录][目标文件或目录]
```

或

```
rcp [-pr][源文件或目录...][目标文件]
```

参数：

-p 保留源文件或目录的属性，包括拥有者，所属群组，权限与时间。

-r 递归处理，将指定目录下的文件与子目录一并处理。

实例

使用rcp指令复制远程文件到本地进行保存。

设本地主机当前账户为rootlocal，远程主机账户为root，要将远程主机（218.6.132.5）主目录下的文件"testfile"复制到本地目录"test"中，则输入如下命令：

```
rcp root@218.6.132.5:./testfile testfile  #复制远程文件到本地
rcp root@218.6.132.5:home/rootlocal/testfile testfile
#要求当前登录账户cmd 登录到远程主机
rcp 218.6.132.5:./testfile testfile
```

注意：指令"rcp"执行以后不会有返回信息，仅需要在目录"test"下查看是否存在文件"testfile"。若存在，则表示远程复制操作成功，否则远程复制操作失败。

Linux rm命令

Linux rm命令用于删除一个文件或者目录。

语法

```
rm [options] name...
```

参数：

- -i 删除前逐一询问确认。
- -f 即使原档案属性设为唯读，亦直接删除，无需逐一确认。
- -r 将目录及以下之档案亦逐一删除。

实例

删除文件可以直接使用rm命令，若删除目录则必须配合选项"-r"，例如：

```
# rm  test.txt
rm：是否删除 一般文件 "test.txt"? y
# rm  homework
rm: 无法删除目录"homework": 是一个目录
# rm  -r  homework
rm：是否删除 目录 "homework"? y
```

删除当前目录下的所有文件及目录，命令行为：

```
rm  -r  *
```

文件一旦通过rm命令删除，则无法恢复，所以必须格外小心地使用该命令。

Linux slocate命令

Linux slocate命令查找文件或目录。

slocate本身具有一个数据库，里面存放了系统中文件与目录的相关信息。

语法

```
slocate [-u][--help][--version][-d <目录>][查找的文件]
```

参数：

- -d<目录>或--database=<目录> 指定数据库所在的目录。
- -u 更新slocate数据库。
- --help 显示帮助。
- --version 显示版本信息。

实例

使用指令"slocate"显示文件名中含有关键字"fdisk"的文件路径信息，输入如下命令：

```
$ slocate fdisk #显示文件名中含有fdisk关键字的文件的路径信息
```

执行以上命令后，指令执行的输出信息如下：

```
$ slocate fdisk #显示文件名中含有fdisk 关键字的文件的路径信息
/root/cfdisk        #搜索到的文件路径列表
/root/fdisk
/root/sfdisk
/usr/include/grub/ieee1275/ofdisk.h
/usr/share/doc/util-Linux/README.cfdisk
/usr/share/doc/util-Linux/README.fdisk.gz
/usr/share/doc/util-Linux/examples/sfdisk.examples.gz
```

Linux split命令

Linux split命令用于将一个文件分割成数个。

该指令将大文件分割成较小的文件，在默认情况下将按照每1000行切割成一个小文件。

语法

```
split [--help][--version][-<行数>][-b <字节>][-C <字节>][-l <行数>][要切割的文件][输出文件名]
```

参数说明：

- -<行数> : 指定每多少行切成一个小文件
- -b<字节> : 指定每多少字节切成一个小文件
- --help : 在线帮助
- --version : 显示版本信息
- -C<字节> : 与参数"-b"相似，但是在切 割时将尽量维持每行的完整性
- [输出文件名] : 设置切割后文件的前置文件名， split会自动在前置文件名后再加上编号

实例

使用指令"split"将文件"README"每6行切割成一个文件，输入如下命令：

```
$ split -6 README       #将README文件每六行分割成一个文件
```

以上命令执行后，指令"split"会将原来的大文件"README"切割成多个以"x"开头的小文件。而在这些小文件中，每个文件都只有6行内容。

使用指令"ls"查看当前目录结构，如下所示：

```
$ ls                                #执行ls指令
#获得当前目录结构
README xaa xad xag xab xae xah xac xaf xai
```

Linux tee命令

Linux tee命令用于读取标准输入的数据，并将其内容输出成文件。

tee指令会从标准输入设备读取数据，将其内容输出到标准输出设备，同时保存成文件。

语法

```
tee [-ai][--help][--version][文件...]
```

参数：

- -a或--append 　附加到既有文件的后面，而非覆盖它．
- -i或--ignore-interrupts 　忽略中断信号。
- --help 　在线帮助。
- --version 　显示版本信息。

实例

使用指令"tee"将用户输入的数据同时保存到文件"file1"和"file2"中，输入如下命令：

```
$ tee file1 file2                   #在两个文件中复制内容
```

以上命令执行后，将提示用户输入需要保存到文件的数据，如下所示：

```
My Linux                         #提示用户输入数据
My Linux                         #输出数据，进行输出反馈
```

此时，可以分别打开文件"file1"和"file2"，查看其内容是否均是"My Linux"即可判断指令"tee"是否执行成功。

Linux tmpwatch命令

Linux tmpwatch命令用于删除暂存文件。

执行tmpwatch指令可删除不必要的暂存文件，您可以设置文件超期时间，单位以小时计算。

语法

```
tmpwatch [-afqv][--test][超期时间][目录...]
```

参数：

- -a或--all 　删除任何类型的文件。
- -f或--force 　强制删除文件或目录，其效果类似rm指令的"-f"参数。
- -q或--quiet 　不显示指令执行过程。
- -v或--verbose 　详细显示指令执行过程。
- -test 　仅作测试，并不真的删除文件或目录。

实例

使用指令"tmpwatch"删除目录"/tmp"中超过一天未使用的文件，输入如下命令：

```
$ tmpwatch 24 /tmp/ #删除/tmp目录中超过一天未使用的文件
```

以上命令执行后，其执行结果如下所示：

```
removing directctmp/orbit-tom if not empty
```

注意：该指令需要root权限，因此在使用tmpwatch命令前应该使用su命令切换用户。切换管理权限操作如下所示：

```
$ su                               #切换到root用户
口令：**********                   #输入用户密码
```

Linux touch命令

Linux touch命令用于修改文件或者目录的时间属性，包括存取时间和更改时间。若文件不存在，系统会建立一个新的文件。

ls -l 可以显示档案的时间记录。

语法

```
touch [-acfm][-d<日期时间>][-r<参考文件或目录>] [-t<日期时间>][--help][--version][文件或目录…]
```

- 参数说明：
- a 改变档案的读取时间记录。
- m 改变档案的修改时间记录。
- c 假如目的档案不存在，不会建立新的档案。与 --no-create 的效果一样。
- f 不使用，是为了与其他 unix 系统的相容性而保留。
- r 使用参考档的时间记录，与 --file 的效果一样。
- d 设定时间与日期，可以使用各种不同的格式。
- t 设定档案的时间记录，格式与 date 指令相同。
- --no-create 不会建立新档案。
- --help 列出指令格式。
- --version 列出版本讯息。

实例

使用指令"touch"修改文件"testfile"的时间属性为当前系统时间，输入如下命令：

```
$ touch testfile                #修改文件的时间属性
```

首先，使用ls命令查看testfile文件的属性，如下所示：

```
$ ls -l testfile                #查看文件的时间属性
#原来文件的修改时间为16:09
-rw-r--r-- 1 hdd hdd 55 2011-08-22 16:09 testfile
```

执行指令"touch"修改文件属性以后，并再次查看该文件的时间属性，如下所示：

```
$ touch testfile                #修改文件时间属性为当前系统时间
$ ls -l testfile                #查看文件的时间属性
#修改后文件的时间属性为当前系统时间
-rw-r--r-- 1 hdd hdd 55 2011-08-22 19:53 testfile
```

使用指令"touch"时，如果指定的文件不存在，则将创建一个新的空白文件。例如，在当前目录下，使用该指令创建一个空白文件"file"，输入如下命令：

```
$ touch file            #创建一个名为“file”的新的空白文件
```

Linux umask命令

Linux umask命令指定在建立文件时预设的权限掩码。

umask可用来设定[权限掩码]。[权限掩码]是由3个八进制的数字所组成，将现有的存取权限减掉权限掩码后，即可产生建立文件时预设的权限。

语法

```
umask [-S][权限掩码]
```

参数说明：

-S　以文字的方式来表示权限掩码。

实例

使用指令"umask"查看当前权限掩码，则输入下面的命令：

```
$ umask                         #获取当前权限掩码
```

执行上面的指令后，输出信息如下：

```
0022
```

接下来，使用指令"mkdir"创建一个目录，并使用指令"ls"获取该目录的详细信息，输入命令如下：

```
$ mkdir test1                       #创建目录
$ ls –d –l test1/                   #显示目录的详细信息
```

执行上面的命令后，将显示新创建目录的详细信息，如下所示：

```
drwxr-xr-x 2 rootlocal rootlocal 4096 2011-9-19 21:46 test1/
```

注意：在上面的输出信息中，"drwxr-xr-x"="777-022=755"。

Linux which命令

Linux which命令用于查找文件。

which指令会在环境变量$PATH设置的目录里查找符合条件的文件。

语法

```
which [文件...]
```

参数：

- -n<文件名长度>　指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。
- -p<文件名长度>　与-n参数相同，但此处的<文件名长度>包括了文件的路径。
- -w　指定输出时栏位的宽度。
- -V　显示版本信息。

实例

使用指令"which"查看指令"bash"的绝对路径，输入如下命令：

```
$ which bash
```

上面的指令执行后，输出信息如下所示：

```
/bin/bash                     #bash可执行程序的绝对路径
```

Linux cp命令

Linux cp命令主要用于复制文件或目录。

语法

```
cp [options] source dest
```

或

```
cp [options] source... directory
```

参数说明：

- -a：此选项通常在复制目录时使用，它保留链接、文件属性，并复制目录下的所有内容。其作用等于dpR参数组合。
- -d：复制时保留链接。这里所说的链接相当于Windows系统中的快捷方式。
- -f：覆盖已经存在的目标文件而不给出提示。
- -i：与-f选项相反，在覆盖目标文件之前给出提示，要求用户确认是否覆盖，回答"y"时目标文件将被覆盖。
- -p：除复制文件的内容外，还把修改时间和访问权限也复制到新文件中。
- -r：若给出的源文件是一个目录文件，此时将复制该目录下所有的子目录和文件。
- -l：不复制文件，只是生成链接文件。

实例

使用指令"cp"将当前目录"test/"下的所有文件复制到新目录"newtest"下，输入如下命令：

```
$ cp –r test/ newtest
```

注意：用户使用该指令复制目录时，必须使用参数"-r"或者"-R"。

Linux mcopy命令

Linux mcopy命令用来复制 MSDOS 格式文件到 Linux 中，或是由 Linux 中复制 MSDOS 文件到磁片上。

mcopy 可复制单一的文件到所指定的文件名称，或是复制数个文件到所指定的目录之中。来源与目的文件可为 MSDOS 或是 Linux 文件。

mcopy指令是一种mtools工具指令，可以在DOS系统中复制文件或者在DOS与Linux操作系统之间进行文件复制。

语法

```
mcopy [-bnmpQt/][源文件][目标文件或目录]
```

参数：

- b 批处理模式。这是为大量的文件复制进行最佳化的选项,但是当在复制文件过程中产生 crash 时，会有安全性的问题产生。/ 递回的复制。包含目录所含文件与其下所有子目录中的文件。
- -n 覆盖其他文件时，不需要进行确认而直接覆盖
- m 将源文件修改时间设置为目标文件的修改时间。
- p 将源文件的属性设置为目标文件的属性。
- Q 当复制多个文件产生错误时，尽快结束程序。
- t 转换为文本文件。
- o 在覆盖 MSDOS 文件时不会出现警示讯息。

实例

将 A 盘根目录中的 autoexec.bat 复制到目前工作目录之下：

```
mcopy a:autoexec.bat .
```

当复制的内容包括子目录和文件时，必须使用参数"-/"递归操作，因此该命令为：

```
mcopy -/ A:\*
```

执行该命令前先使用mdir 命令查看原来的目录结构，执行mcopy 之后可使用ls 命令查看复制之后Linux系统中的文件结构，结果如下：

```
cmd@cmd-desktop:~$ mdir -/ a:\* #查看A 盘中的文件
Volume in drive A has no label #加载信息
Volume Serial Number is 13D2~055C
Directory for A:/ #以下为目录信息
#文件名目录大小 修改时间
./TEST <DIR> 2009-09-23 16:59
AUTORUN.INF 265 2009-09-23 16:53
AUTORUN.BAT 43 2009-09-23 16:56
3 files 308 bytes #统计总大小
724 325 bytes free #剩余空间
cmd@cmd-desktop:~$ mcopy -/ A:\* #将A盘上的所有文件复制到当前工作目录
cmd@cmd-desktop:~$ls
TEST AUTORUN.INF AUTORUN.BAT #A盘中的内容复制到Linux文件系统结构中
```

Linux mshowfat命令

Linux mshowfat命令用于显示MS-DOS文件在FAT中的记录。

mshowfat为mtools工具指令，可显示MS-DOS文件在FAT中的记录编号。

语法

```
mshowfat [文件...]
```

参数说明：

[文件...]：执行操作的文件相对路径或者绝对路径

实例

使用指令mshowfat查看文件"autorun.bat"的FAT信息，输入如下命令：

```
$ mshowfat autorun.bat
```

以上命令执行后，文件"autorun.bat"的FAT相关信息将会被显示出来。

注意：执行操作的文件必须是DOS文件系统下的文件。

Linux rhmask命令

Linux rhmask命令用于对文件进行加密和解密操作。

执行rhmask指令可制作加密过的文件，方便用户在公开的网络上传输该文件，而不至于被任意盗用。

语法

```
rhmask [加密文件][输出文件] 或 rhmask [-d][加密文件][源文件][输出文件]
```

参数：

- -d 　产生加密过的文件。

实例

使用指令"rhmask"将加密文件"code.txt"进行加密后，另存为输出文件"demo.txt"，输入如下命令：

```
$ rhmask code.txt demo.txt
```

以上命令执行后，文件"code.txt"将被加密后，另存为已经加密的文件"demo.txt"。

注意：该指令有两种语法，用户可以有选择性地进行使用即可。

Linux whereis命令

Linux whereis命令用于查找文件。

该指令会在特定目录中查找符合条件的文件。这些文件应属于原始代码、二进制文件，或是帮助文件。

该指令只能用于查找二进制文件、源代码文件和man手册页，一般文件的定位需使用locate命令。

语法

```
whereis [-bfmsu][-B <目录>...][-M <目录>...][-S <目录>...][文件...]
```

参数：

- -b 只查找二进制文件。 *-B<目录>* 只在设置的目录下查找二进制文件。 -f 不显示文件名前的路径名称。 *-m* 只查找说明文件。 -M<目录> 只在设置的目录下查找说明文件。 *-s* 只查找原始代码文件。 -S<目录> 只在设置的目录下查找原始代码文件。* -u 查找不包含指定类型的文件。

实例

使用指令"whereis"查看指令"bash"的位置，输入如下命令：

```
$ whereis bash
```

上面的指令执行后，输出信息如下所示：

```
bash:/bin/bash/etc/bash.bashrc/usr/share/man/man1/bash.1.gz
```

注意：以上输出信息从左至右分别为查询的程序名、bash路径、bash的man 手册页路径。

如果用户需要单独查询二进制文件或帮助文件，可使用如下命令：

```
$ whereis -b bash
$ whereis -m bash
```

输出信息如下：

```
$ whereis -b bash               #显示bash 命令的二进制程序
bash: /bin/bash /etc/bash.bashrc /usr/share/bash    # bash命令的二进制程序的地址
$ whereis -m bash               #显示bash 命令的帮助文件
bash: /usr/share/man/man1/bash.1.gz  #bash命令的帮助文件地址
```

Linux scp命令

Linux scp命令用于Linux之间复制文件和目录。

scp是 secure copy的缩写, scp是linux系统下基于ssh登陆进行安全的远程文件拷贝命令。

语法

```
scp [-1246BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
[-l limit] [-o ssh_option] [-P port] [-S program]
[[user@]host1:]file1 [...] [[user@]host2:]file2
```

简易写法:

```
scp [可选参数] file_source file_target
```

参数说明：

- -1： 强制scp命令使用协议ssh1
- -2： 强制scp命令使用协议ssh2
- -4： 强制scp命令只使用IPv4寻址
- -6： 强制scp命令只使用IPv6寻址
- -B： 使用批处理模式（传输过程中不询问传输口令或短语）
- -C： 允许压缩。（将-C标志传递给ssh，从而打开压缩功能）
- -p：保留原文件的修改时间，访问时间和访问权限。
- -q： 不显示传输进度条。
- -r： 递归复制整个目录。
- -v：详细方式显示输出。scp和ssh(1)会显示出整个过程的调试信息。这些信息用于调试连接，验证和配置问题。
- -c cipher： 以cipher将数据传输进行加密，这个选项将直接传递给ssh。
- -F ssh_config： 指定一个替代的ssh配置文件，此参数直接传递给ssh。
- -i identity_file： 从指定文件中读取传输时使用的密钥文件，此参数直接传递给ssh。
- -l limit： 限定用户所能使用的带宽，以Kbit/s为单位。
- -o ssh_option： 如果习惯于使用ssh_config(5)中的参数传递方式，
- -P port：注意是大写的P, port是指定数据传输用到的端口号
- -S program： 指定加密传输时所使用的程序。此程序必须能够理解ssh(1)的选项。

实例

1、从本地复制到远程

命令格式：

```
scp local_file remote_username@remote_ip:remote_folder
或者
scp local_file remote_username@remote_ip:remote_file
或者
scp local_file remote_ip:remote_folder
或者
scp local_file remote_ip:remote_file
```

- 第1,2个指定了用户名，命令执行后需要再输入密码，第1个仅指定了远程的目录，文件名字不变，第2个指定了文件名；
- 第3,4个没有指定用户名，命令执行后需要输入用户名和密码，第3个仅指定了远程的目录，文件名字不变，第4个指定了文件名；

应用实例：

```
scp /home/space/music/1.mp3 root@www.w3cschool.cc:/home/root/others/music
scp /home/space/music/1.mp3 root@www.w3cschool.cc:/home/root/others/music/001.mp3
scp /home/space/music/1.mp3 www.w3cschool.cc:/home/root/others/music
scp /home/space/music/1.mp3 www.w3cschool.cc:/home/root/others/music/001.mp3
```

复制目录命令格式：

```
scp -r local_folder remote_username@remote_ip:remote_folder
或者
scp -r local_folder remote_ip:remote_folder
```

- 第1个指定了用户名，命令执行后需要再输入密码；
- 第2个没有指定用户名，命令执行后需要输入用户名和密码；

应用实例：

```
scp -r /home/space/music/ root@www.w3cschool.cc:/home/root/others/
scp -r /home/space/music/ www.w3cschool.cc:/home/root/others/
```

上面命令将本地 music 目录复制到远程 others 目录下。

2、从远程复制到本地

从远程复制到本地，只要将从本地复制到远程的命令的后2个参数调换顺序即可，如下实例

应用实例：

```
scp root@www.w3cschool.cc:/home/root/others/music /home/space/music/1.mp3
scp -r www.w3cschool.cc:/home/root/others/ /home/space/music/
```

说明

1.如果远程服务器防火墙有为scp命令设置了指定的端口，我们需要使用 -p 参数来设置命令的端口号，命令格式如下：

```
#scp命令使用端口号 4588
scp -p 4588 remote@www.w3cschool.cc:/usr/local/sin.sh /home/administrator
```

2.使用scp命令要确保使用的用户具有可读取远程服务器相应文件的权限，否则scp命令是无法起作用的。

Linux awk 命令

AWK是一种处理文本文件的语言，是一个强大的文本分析工具。

之所以叫AWK是因为其取了三位创始人 Alfred Aho，Peter Weinberger, 和 Brian Kernighan 的Family Name的首字符。

语法

```
awk [选项参数] 'script' var=value file(s)
或
awk [选项参数] -f scriptfile var=value file(s)
```

选项参数说明：

- -F fs or --field-separator fs
  指定输入文件折分隔符，fs是一个字符串或者是一个正则表达式，如-F:。
- -v var=value or --asign var=value
  赋值一个用户定义变量。
- -f scripfile or --file scriptfile
  从脚本文件中读取awk命令。
- -mf nnn and -mr nnn
  对nnn值设置内在限制，-mf选项限制分配给nnn的最大块数目；-mr选项限制记录的最大数目。这两个功能是Bell实验室版awk的扩展功能，在标准awk中不适用。
- -W compact or --compat, -W traditional or --traditional
  在兼容模式下运行awk。所以gawk的行为和标准的awk完全一样，所有的awk扩展都被忽略。
- -W copyleft or --copyleft, -W copyright or --copyright
  打印简短的版权信息。
- -W help or --help, -W usage or --usage
  打印全部awk选项和每个选项的简短说明。
- -W lint or --lint
  打印不能向传统unix平台移植的结构的警告。
- -W lint-old or --lint-old
  打印关于不能向传统unix平台移植的结构的警告。
- -W posix
  打开兼容模式。但有以下限制，不识别：/x、函数关键字、func、换码序列以及当fs是一个空格时，将新行作为一个域分隔符；操作符和=不能代替^和^=；fflush无效。
- -W re-interval or --re-inerval
  允许间隔正则表达式的使用，参考(grep中的Posix字符类)，如括号表达式[[:alpha:]]。

- -W source program-text or --source program-text
  使用program-text作为源代码，可与-f命令混用。
- -W version or --version
  打印bug报告信息的版本。

基本用法

log.txt文本内容如下：

```
2 this is a test
3 Are you like awk
This's a test
10 There are orange,apple,mongo
```

用法一：

```
awk '{[pattern] action}' {filenames}   # 行匹配语句 awk '' 只能用单引号
```

实例：

```
# 每行按空格或TAB分割，输出文本中的1、4项
 $ awk '{print $1,$4}' log.txt
 ---------------------------------------------
 2 a
 3 like
 This's
 10 orange,apple,mongo
 # 格式化输出
 $ awk '{printf "%-8s %-10s\n",$1,$4}' log.txt
 ---------------------------------------------
 2        a
 3        like
 This's
 10       orange,apple,mongo
```

用法二：

```
awk -F  #-F相当于内置变量FS, 指定分割字符
```

实例：

```
# 使用","分割
 $  awk -F, '{print $1,$2}'   log.txt
 ---------------------------------------------
 2 this is a test
 3 Are you like awk
 This's a test
 10 There are orange apple
 # 或者使用内建变量
 $ awk 'BEGIN{FS=","} {print $1,$2}'     log.txt
 ---------------------------------------------
 2 this is a test
 3 Are you like awk
 This's a test
 10 There are orange apple
 # 使用多个分隔符.先使用空格分割, 然后对分割结果再使用","分割
 $ awk -F '[ ,]'  '{print $1,$2,$5}'   log.txt
 ---------------------------------------------
 2 this test
 3 Are awk
 This's a
 10 There apple
```

用法三：

```
awk -v  # 设置变量
```

实例：

```
 $ awk -va=1 '{print $1,$1+a}' log.txt
 ---------------------------------------------
 2 3
 3 4
 This's 1
 10 11
 $ awk -va=1 -vb=s '{print $1,$1+a,$1b}' log.txt
 ---------------------------------------------
 2 3 2s
 3 4 3s
 This's 1 This'ss
 10 11 10s
```

用法四：

```
awk -f {awk脚本} {文件名}
```

实例：

```
 $ awk -f cal.awk log.txt
```

运算符

运算符	描述
= += -= = /= %= ^= *=	赋值
?:	C条件表达式
||	逻辑或
&&	逻辑与
~ ~!	匹配正则表达式和不匹配正则表达式
< <= > >= != ==	关系运算符
空格	连接
+ -	加，减
* / &	乘，除与求余
+ - !	一元加，减和逻辑非
^ *	求幂
++ --	增加或减少，作为前缀或后缀
$	字段引用
in	数组成员

过滤第一列大于2的行

```
$ awk '$1>2' log.txt    #命令
#输出
3 Are you like awk
This's a test
10 There are orange,apple,mongo
```

过滤第一列等于2的行

```
$ awk '$1==2 {print $1,$3}' log.txt    #命令
#输出
2 is
```

过滤第一列大于2并且第二列等于'Are'的行

```
$ awk '$1>2 && $2=="Are" {print $1,$2,$3}' log.txt    #命令
#输出
3 Are you
```

内建变量

变量	描述
$n	当前记录的第n个字段，字段间由FS分隔
$0	完整的输入记录
ARGC	命令行参数的数目
ARGIND	命令行中当前文件的位置(从0开始算)
ARGV	包含命令行参数的数组
CONVFMT	数字转换格式(默认值为%.6g)ENVIRON环境变量关联数组
ERRNO	最后一个系统错误的描述
FIELDWIDTHS	字段宽度列表(用空格键分隔)
FILENAME	当前文件名
FNR	同NR，但相对于当前文件
FS	字段分隔符(默认是任何空格)
IGNORECASE	如果为真，则进行忽略大小写的匹配
NF	当前记录中的字段数
NR	当前记录数
OFMT	数字的输出格式(默认值是%.6g)
OFS	输出字段分隔符(默认值是一个空格)
ORS	输出记录分隔符(默认值是一个换行符)
RLENGTH	由match函数所匹配的字符串的长度
RS	记录分隔符(默认是一个换行符)
RSTART	由match函数所匹配的字符串的第一个位置
SUBSEP	数组下标分隔符(默认值是/034)

```
$ awk 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR","FS",
FILENAME ARGC  FNR   FS   NF   NR  OFS  ORS   RS
---------------------------------------------
log.txt    2    1         5    1
log.txt    2    2         5    2
log.txt    2    3         3    3
log.txt    2    4         4    4
$ awk -F\' 'BEGIN{printf "%4s %4s %4s %4s %4s %4s %4s %4s %4s\n","FILENAME","ARGC","FNR",
FILENAME ARGC  FNR   FS   NF   NR  OFS  ORS   RS
---------------------------------------------
log.txt    2    1    '    1    1
log.txt    2    2    '    1    2
log.txt    2    3    '    2    3
log.txt    2    4    '    1    4
# 输出顺序号 NR, 匹配文本行号
$ awk '{print NR,FNR,$1,$2,$3}' log.txt
---------------------------------------------
1 1 2 this is
2 2 3 Are you
3 3 This's a test
4 4 10 There are
# 指定输出分割符
$  awk '{print $1,$2,$5}' OFS=" $ "  log.txt
---------------------------------------------
2 $ this $ test
3 $ Are $ awk
This's $ a $
10 $ There $
```

使用正则，字符串匹配

```
# 输出第二列包含 "th"，并打印第二列与第四列
$ awk '$2 ~ /th/ {print $2,$4}' log.txt
---------------------------------------------
this a
```

~ 表示模式开始。// 中是模式。

```
# 输出包含"re" 的行
$ awk '/re/ ' log.txt
---------------------------------------------
3 Are you like awk
10 There are orange,apple,mongo
```

忽略大小写

```
$ awk 'BEGIN{IGNORECASE=1} /this/' log.txt
---------------------------------------------
2 this is a test
This's a test
```

模式取反

```
$ awk '$2 !~ /th/ {print $2,$4}' log.txt
---------------------------------------------
Are like
a
There orange,apple,mongo
$ awk '!/th/ {print $2,$4}' log.txt
---------------------------------------------
Are like
a
There orange,apple,mongo
```

awk脚本

关于awk脚本，我们需要注意两个关键词BEGIN和END。

- BEGIN{ 这里面放的是执行前的语句 }
- END {这里面放的是处理完所有的行后要执行的语句 }
- {这里面放的是处理每一行时要执行的语句}

假设有这么一个文件（学生成绩表）：

```
$ cat score.txt
Marry   2143 78 84 77
Jack    2321 66 78 45
Tom     2122 48 77 71
Mike    2537 87 97 95
Bob     2415 40 57 62
```

我们的awk脚本如下：

```
$ cat cal.awk
#!/bin/awk -f
#运行前
BEGIN {
    math = 0
    english = 0
    computer = 0

    printf "NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL\n"
    printf "---------------------------------------------\n"
}
#运行中
{
    math+=$3
    english+=$4
    computer+=$5
    printf "%-6s %-6s %4d %8d %8d %8d\n", $1, $2, $3,$4,$5, $3+$4+$5
}
#运行后
END {
    printf "---------------------------------------------\n"
    printf "  TOTAL:%10d %8d %8d \n", math, english, computer
    printf "AVERAGE:%10.2f %8.2f %8.2f\n", math/NR, english/NR, computer/NR
}
```

我们来看一下执行结果：

```
$ awk -f cal.awk score.txt
NAME    NO.   MATH  ENGLISH  COMPUTER   TOTAL
---------------------------------------------
Marry  2143     78       84       77      239
Jack   2321     66       78       45      189
Tom    2122     48       77       71      196
Mike   2537     87       97       95      279
Bob    2415     40       57       62      159
---------------------------------------------
  TOTAL:       319      393      350
AVERAGE:     63.80    78.60    70.00
```

另外一些实例

AWK的hello world程序为：

```
BEGIN { print "Hello, world!" }
```

计算文件大小

```
$ ls -l *.txt | awk '{sum+=$6} END {print sum}'
--------------------------------------------------
666581
```

从文件中找出长度大于80的行

```
awk 'lenght>80' log.txt
```

打印九九乘法表

```
seq 9 | sed 'H;g' | awk -v RS='' '{for(i=1;i<=NF;i++)printf("%dx%d=%d%s", i, NR, i*NR, i=
```

更多详细内容可以查看 AWK 官方手册：http://www.gnu.org/software/gawk/manual/gawk.html

Linux命令大全 - 文档编辑

col	colrm	comm	csplit
ed	egrep	ex	fgrep
fmt	fold	grep	ispell
jed	joe	join	look
mtype	pico	rgrep	sed
sort	spell	tr	expr
uniq	wc		

Linux col命令

Linux col命令用于过滤控制字符。

在许多UNIX说明文件里，都有RLF控制字符。当我们运用shell特殊字符">"和">>"，把说明文件的内容输出成纯文本文件时，控制字符会变成乱码，col指令则能有效滤除这些控制字符。

语法

```
col [-bfx][-l<缓冲区列数>]
```

参数：

- -b 过滤掉所有的控制字符，包括RLF和HRLF。
- -f 滤除RLF字符，但允许将HRLF字符呈现出来。
- -x 以多个空格字符来表示跳格字符。
- -l<缓冲区列数> 预设的内存缓冲区有128列，您可以自行指定缓冲区的大小。

实例

下面以 man 命令帮助文档为例，讲解col 命令的使用。

将man 命令的帮助文档保存为man_help，使用-b 参数过滤所有控制字符。在终端中使用如下命令：

```
man man | col-b > man_help
```

注:其中"|"用于建立管道，把man命令的输出结果转为col命令的输入数据。

Linux colrm命令

Linux colrm命令用于滤掉指定的行。

colrm指令从标准输入设备读取书记，转而输出到标准输出设备。如果不加任何参数，则该指令不会过滤任何一行。

语法

```
colrm [开始行数编号<结束行数编号>]</p>

<p><b>参数说明：</b></p>
<ul>
<li>开始行数编号： 指定要删除的列的起始编号。</li>
<li>结束行数编号： 指定要删除的列的结束编号，有时候这个参数可以省略。</li>
</ul>
<h3>实例</h3>
<p>不带任何参数时该命令不会删除任何列：</p>
<pre>
colrm
```

按回车键后，光标将在第一行闪烁，等待标准输入，此时输入字符，如"Hello Linux！"，再按回车键后第二行将出现与第一行相同内容，此时按Ctrl+C组合键可以退出。终端中显示的内容如下所示：

```
cmd@hdd-desktop:~$ colrm
Hello Linux! #输入Hello Linux！字符串
Hello Linux! #输出刚才输入的字符串Hello Linux！
```

如想要删除第4 列之后的所有内容，可以使用如下命令：

```
colrm 4
```

类似于上例，此时标准输入等待输入，用户输入字符串按回车键后，将输出如下结果：

```
cmd@hdd-desktop:~$ colrm 4
Hello Linux! #输入Hello Linux！字符串
Hel #输出删除了第4列以后所有内容的字符串
```

删除指定列的内容。如删除第4列到第6列的内容，可使用如下命令：

```
colrm 4 6
```

输出的结果如下：

```
cmd@hdd-desktop:~$ colrm 4 6
Hello Linux! #输入Hello Linux！字符串
HelLinux! #输出删除了从第4列到第6列字符的字符串
```

Linux comm命令

Linux comm命令用于比较两个已排过序的文件。

这项指令会一列列地比较两个已排序文件的差异，并将其结果显示出来，如果没有指定任何参数，则会把结果分成3行显示：第1行仅是在第1个文件中出现过的列，第2行是仅在第2个文件中出现过的列，第3行则是在第1与第2个文件里都出现过的列。若给予的文件名称为"-"，则comm指令会从标准输入设备读取数据。

语法

```
comm [-123][--help][--version][第1个文件][第2个文件]
```

参数：

- -1 不显示只在第1个文件里出现过的列。
- -2 不显示只在第2个文件里出现过的列。
- -3 不显示只在第1和第2个文件里出现过的列。
- --help 在线帮助。
- --version 显示版本信息。

实例

aaa.txt 与 bbb.txt 的文件内容如下：

```
[root@localhost text]# cat aaa.txt
aaa
bbb
ccc
ddd
eee
111
222
[root@localhost text]# cat bbb.txt
bbb
ccc
aaa
hhh
ttt
jjj
```

```
<p>执行 comm 命令输出结果如下：</p>
[root@localhost text]# comm aaa.txt bbb.txt
aaa
                bbb
                ccc
        aaa
ddd
eee
111
222
        hhh
        ttt
        jjj
第一列  第二列  第三列
```

输出的第一列只包含在aaa.txt中出现的行，第二列包含在bbb.txt中出现的行，第三列包含在aaa.txt和bbb.txt中相同的行。各列是以制表符（\t）作为定界符。

Linux csplit命令

Linux csplit命令用于分割文件。

将文件依照指定的范本样式予以切割后，分别保存成名称为xx00,xx01,xx02...的文件。若给予的文件名称为"-"，则csplit指令会从标准输入设备读取数据。

语法

```
csplit [-kqsz][-b<输出格式>][-f<输出字首字符串>]
[-n<输出文件名位数>][--help][--version][文件][范本样式...]
```

参数：

- -b<输出格式>或--suffix-format=<输出格式> 预设的输出格式其文件名称为xx00,xx01...等，您可以通过改变<输出格式>来改变输出的文件名。
- -f<输出字首字符串>或--prefix=<输出字首字符串> 预设的输出字首字符串其文件名为xx00,xx01...等，如果你指定输出字首字符串为"hello"，则输出的文件名称会变成hello00,hello01...等。
- -k或--keep-files 保留文件，就算发生错误或中断执行，也不能删除已经输出保存的文件。
- -n<输出文件名位数>或--digits=<输出文件名位数> 预设的输出文件名位数其文件名称为xx00,xx01...等，如果你指定输出文件名位数为"3"，则输出的文件名称会变成xx000,xx001...等。
- -q或-s或--quiet或--silent 不显示指令执行过程。
- -z或--elide-empty-files 删除长度为0 Byte文件。
- --help 在线帮助。
- --version 显示版本信息。

实例

将文本文件testfile以第 2 行为分界点切割成两份，使用如下命令：

```
csplit testfile 2
```

testfile文件中的内容如下：

```
$ cat testfile #查看testfile 文件内容
hello Linux!
Linux is a free Unix-type operating system.
This is a Linux testfile!
Linux
```

使用csplit命令，输出结果如下：

```
$ csplit testfile 2
13 #xx00文件字符个数
76 #xx01文件字符个数
```

其中第1 行是第一个文件xx00的字符个数，同样，第2 行为第二个文件xx01的字符个数。同时，在testfile 的同目录下将生成两个文件，文件名分别为xx00、xx01，xx00 中的内容为：

```
$ cat xx00 #查看分割后的xx00文件内容
hello Linux! #testfile文件第1行的内容
```

xx01 中的内容为：

```
$ cat xx01 #查看分割后的xx01文件内容
Linux is a free Unix-type operating system. #testfile文件第2行以后的内容
This is a Linux testfile!
Linux
```

Linux ed命令

Linux ed命令是文本编辑器，用于文本编辑。

ed是Linux中功能最简单的文本编辑程序，一次仅能编辑一行而非全屏幕方式的操作。

ed命令并不是一个常用的命令，一般使用比较多的是vi 指令。但ed文本编辑器对于编辑大文件或对于在shell脚本程序中进行文本编辑很有用。

语法

```
ed [-][-Gs][-p<字符串>][--help][--version][文件]
```

参数：

- -G或--traditional 提供回兼容的功能。
- -p<字符串> 指定ed在command mode的提示字符。
- -s,-,--quiet或--silent 不执行开启文件时的检查功能。
- --help 显示帮助。
- --version 显示版本信息。

实例

以下是一个 Linux ed 完整实例解析：

```
$ ed                <- 激活 ed 命令
a                   <- 告诉 ed 我要编辑新文件
My name is Titan. <- 输入第一行内容
And I love Perl very much. <- 输入第二行内容
.                   <- 返回 ed 的命令行状态
i                   <- 告诉 ed 我要在最后一行之前插入内容
I am 24\.            <- 将“I am 24.”插入“My name is Titan.”和“And I love Perl very much.”之间
.                   <- 返回 ed 的命令行状态
c                   <- 告诉 ed 我要替换最后一行输入内容
I am 24 years old. <- 将“I am 24.”替换成“I am 24 years old.”（注意：这里替换的是最后输的内容）
.                   <- 返回 ed 的命令行状态
w readme.text       <- 将文件命名为“readme.text”并保存（注意：如果是编辑已经存在的文件，只需要敲入 w .
q                   <- 完全退出 ed 编辑器
```

这是文件的内容是：

```
$ cat readme.text
My name is Titan.
I am 24 years old.
And I love Perl vrey much.
```

Linux egrep命令

Linux egrep命令用于在文件内查找指定的字符串。

egrep执行效果与"grep-E"相似，使用的语法及参数可参照grep指令，与grep的不同点在于解读字符串的方法。

egrep是用extended regular expression语法来解读的，而grep则用basic regular expression语法解读，extended regular expression比basic regular expression的表达更规范。

语法

```
egrep [范本模式] [文件或目录]
```

参数说明：

- [范本模式] ：查找的字符串规则。
- [文件或目录] ：查找的目标文件或目录。

实例

显示文件中符合条件的字符。例如，查找当前目录下所有文件中包含字符串"Linux"的文件，可以使用如下命令：

```
egrep Linux *
```

结果如下所示：

```
$ egrep Linux * #查找当前目录下包含字符串“Linux”的文件
testfile:hello Linux! #以下五行为testfile 中包含Linux字符的行
testfile:Linux is a free Unix-type operating system.
testfile:This is a Linux testfile!
testfile:Linux
testfile:Linux
testfile1:helLinux! #以下两行为testfile1中含Linux字符的行
testfile1:This a Linux testfile!
#以下两行为testfile_2 中包含Linux字符的行
testfile_2:Linux is a free unix-type opterating system.
testfile_2:Linux test
xx00:hello Linux! #xx00包含Linux字符的行
xx01:Linux is a free Unix-type operating system. #以下三行为xx01包含Linux字符的行
xx01:This is a Linux testfile!
xx01:Linux
```

Linux ex命令

Linux ex命令用于在Ex模式下启动vim文本编辑器。

ex执行效果如同vi -E，使用语法及参数可参照vi指令，如要从Ex模式回到普通模式，则在vim中输入":vi"或":visual"指令即可。

语法

```
ex [选项][参数]
```

参数说明：

- -b：使用二进制模式编辑文件
- -c 指令：编辑完第一个文件后执行指定的指令
- -d ：编辑多个文件时，显示差异部分
- -m ：不允许修改文件
- -n ：不使用缓存
- -oN：其中 N 为数字
- -r ：列出缓存，并显示恢复信息
- -R ：以只读的方式打开文件
- -s ：不显示任何错误信息
- -V ：显示指令的详细执行过程
- --help ：显示帮助信息
- --version ：显示版本信息

实例

在ex 指令后输入文件名按回车键后，即可进入ex 编辑模式，如编辑testfile文件，使用的命令格式如下：

```
ex testfile
```

输出的信息如下：

```
"testfile" 5L, 95C
```

"testfile"表示文件名，5L表示5 行，95 表示字节数

进入ex 模式。输入"visual"回到正常模式

它的操作与vim 中是一样的，此时如果在":"后输入"visual"后按回车键，将进入到vi 指令全屏界面；如果输入"q"，则退出编辑器。

Linux fgrep命令

本指令相当于执行grep指令加上参数"-F"，详见grep命令说明。

Linux fgrep命令用于查找文件里符合条件的字符串。

语法

```
fgrep [范本样式][文件或目录...]
```

实例

具体使用实例请参考grep命令。

Linux fmt命令

Linux fmt命令用于编排文本文件。

fmt指令会从指定的文件里读取内容，将其依照指定格式重新编排后，输出到标准输出设备。若指定的文件名为"-"，则fmt指令会从标准输入设备读取数据。

语法

```
fmt [-cstu][-p<列起始字符串>][-w<每列字符数>][--help][--version][文件...]
```

参数说明：

- -c或--crown-margin 每段前两列缩排。
- -p<列起始字符串>或-prefix=<列起始字符串> 仅合并含有指定字符串的列，通常运用在程序语言的注解方面。
- -s或--split-only 只拆开字数超出每列字符数的列，但不合并字数不足每列字符数的列。
- -t或--tagged-paragraph 每列前两列缩排，但第1列和第2列的缩排格式不同。
- -u或--uniform-spacing 每个字符之间都以一个空格字符间隔，每个句子之间则两个空格字符分隔。
- -w<每列字符数>或--width=<每列字符数>或-<每列字符数> 设置每列的最大字符数。
- --help 在线帮助。
- --version 显示版本信息。

实例

重排指定文件。如文件testfile共5 行文字，可以通过命令对该文件格式进行重排，其命令为：

```
fmt testfile
```

输出结果如下：

```
$ fmt testfile #重排testfile 文件
hello Linux! Linux is a free Unix-type operating system. This is a
Linux testfile! Linux Linux
```

将文件testfile重新排成85 个字符一行，并在标准输出设备上输出，其命令应该为：

```
fmt -w 85 testfile
```

为了对比，先使用cat 命令查看文件内容：

```
$ cat testfile #查看testfile 文件的内容
hello Linux!
Linux is a free Unix-type operating system.
This is a Linux testfile!
Linux
Linux
```

使用fmt命令重排之后，输出结果如下：

```
$ fmt -w 85 testfile #指定重排宽度为85个字符
hello Linux! Linux is a free Unix-type operating system. This is a Linux testfile!
Linux Linux
```

Linux fold命令

Linux fold命令用于限制文件列宽。

fold指令会从指定的文件里读取内容，将超过限定列宽的列加入增列字符后，输出到标准输出设备。若不指定任何文件名称，或是所给予的文件名为"-"，则fold指令会从标准输入设备读取数据。

语法

```
fold [-bs][-w<每列行数>][--help][--version][文件...]
```

参数：

- -b或--bytes 以Byte为单位计算列宽，而非采用行数编号为单位。
- -s或--spaces 以空格字符作为换列点。
- -w<每列行数>或--width<每列行数> 设置每列的最大行数。
- --help 在线帮助。
- --version 显示版本信息。

实例

将一个名为testfile 的文件的行折叠成宽度为30，可使用如下命令：

```
fold -w 30 testfile
```

为了对比，先将testfile文件输出如下：

```
$ cat testfile #查看testfile 中的内容
Linux networks are becoming more and more common, but
security is often an overlooked
issue. Unfortunately, in today’s environment all networks
are potential hacker targets,
from top-secret military research networks to small home LANs.
Linux Network Security focuses on securing Linux in a
networked environment, where the
security of the entire network needs to be considered
rather than just isolated machines.
It uses a mix of theory and practical techniques to
teach administrators how to install and
use security applications, as well as how the
applications work and why they are necessary.
```

然后使用fold命令折叠显示：

```
$ fold -w 30 testfile #行折叠成宽度为30，显示testfile 文件
Linux networks are becoming mo
re and more common, but securi
ty is often an overlooked issu
e. Unfortunately, in today’s
environment all networks are
potential hacker targets, from
top-secret military research
networks to small home LANs.
Linux Network Security focuses
on securing Linux in a networ
ked environment, where the sec
urity of the entire network ne
eds to be considered rather th
an just isolated machines. It
uses a mix of theory and pract
ical techniques to teach admin
istrators how to install and u
se security applications, as w
ell as how the applications wo
rk and why they are necessary
```

Linux grep命令

Linux grep命令用于查找文件里符合条件的字符串。

grep指令用于查找内容包含指定的范本样式的文件，如果发现某文件的内容符合所指定的范本样式，预设grep指令会把含有范本样式的那一列显示出来。若不指定任何文件名称，或是所给予的文件名为"-"，则grep指令会从标准输入设备读取数据。

语法

```
grep [-abcEFGhHilLnqrsvVwxy][-A<显示列数>][-B<显示列数>][-C<显示列数>][-d<进行动作>][-e<范本样式
```

参数：

- -a或--text 不要忽略二进制的数据。
- -A<显示列数>或--after-context=<显示列数> 除了显示符合范本样式的那一列之外，并显示该列之后的内容。
- -b或--byte-offset 在显示符合范本样式的那一列之前，标示出该列第一个字符的位编号。
- -B<显示列数>或--before-context=<显示列数> 除了显示符合范本样式的那一列之外，并显示该列之前的内容。
- -c或--count 计算符合范本样式的列数。
- -C<显示列数>或--context=<显示列数>或-<显示列数> 除了显示符合范本样式的那一列之外，并显示该列之前后的内容。
- -d<进行动作>或--directories=<进行动作> 当指定要查找的是目录而非文件时，必须使用这项参数，否则grep指令将回报信息并停止动作。
- -e<范本样式>或--regexp=<范本样式> 指定字符串做为查找文件内容的范本样式。
- -E或--extended-regexp 将范本样式为延伸的普通表示法来使用。
- -f<范本文件>或--file=<范本文件> 指定范本文件，其内容含有一个或多个范本样式，让grep查找符合范本条件的文件内容，格式为每列一个范本样式。
- -F或--fixed-regexp 将范本样式视为固定字符串的列表。
- -G或--basic-regexp 将范本样式视为普通的表示法来使用。
- -h或--no-filename 在显示符合范本样式的那一列之前，不标示该列所属的文件名称。
- -H或--with-filename 在显示符合范本样式的那一列之前，表示该列所属的文件名称。
- -i或--ignore-case 忽略字符大小写的差别。
- -l或--file-with-matches 列出文件内容符合指定的范本样式的文件名称。
- -L或--files-without-match 列出文件内容不符合指定的范本样式的文件名称。
- -n或--line-number 在显示符合范本样式的那一列之前，标示出该列的列数编号。
- -q或--quiet或--silent 不显示任何信息。

- -r或--recursive 此参数的效果和指定"-d recurse"参数相同。
- -s或--no-messages 不显示错误信息。
- -v或--revert-match 反转查找。
- -V或--version 显示版本信息。
- -w或--word-regexp 只显示全字符合的列。
- -x或--line-regexp 只显示全列符合的列。
- -y 此参数的效果和指定"-i"参数相同。
- --help 在线帮助。

实例

1、在当前目录中，查找后缀有"test"字样的文件中包含"test"字符串的文件，并打印出该字符串的行。此时，可以使用如下命令：

```
grep test *file
```

结果如下所示：

```
$ grep test test* #查找后缀有“test”的文件包含“test”字符串的文件
testfile1:This a Linux testfile! #列出testfile1 文件中包含test字符的行
testfile_2:This is a linux testfile! #列出testfile_2 文件中包含test字符的行
testfile_2:Linux test #列出testfile_2 文件中包含test字符的行
```

2、以递归的方式查找符合条件的文件。例如，查找指定目录/etc/acpi 及其子目录（如果存在子目录的话）下所有文件中包含字符串"update"的文件，并打印出该字符串所在行的内容，使用的命令为：

```
grep -r update /etc/acpi
```

输出结果如下：

```
$ grep-r update /etc/acpi #以递归的方式查找“etc/acpi”
#下包含“update”的文件
/etc/acpi/ac.d/85-anacron.sh:# (Things like the slocate updatedb cause a lot of IO.)
Rather than
/etc/acpi/resume.d/85-anacron.sh:# (Things like the slocate updatedb cause a lot of
IO.) Rather than
/etc/acpi/events/thinkpad-cmos:action=/usr/sbin/thinkpad-keys--update
```

3、反向查找。前面各个例子是查找并打印出符合条件的行，通过"-v"参数可以打印出不符合条件行的内容。

查找文件名中包含test 的文件中不包含test 的行，此时，使用的命令为：

```
grep -v test*
```

结果如下所示：

```
$ grep-v test* #查找文件名中包含test 的文件中不包含test 的行
testfile1:helLinux!
testfile1:Linis a free Unix-type operating system.
testfile1:Lin
testfile_1:HELLO LINUX!
testfile_1:LINUX IS A FREE UNIX-TYPE OPTERATING SYSTEM.
testfile_1:THIS IS A LINUX TESTFILE!
testfile_2:HELLO LINUX!
testfile_2:Linux is a free unix-type opterating system.
```

Linux ispell命令

Linux ispell命令用于拼写检查程序。

ispell预设会使用/usr/lib/ispell/english.hash字典文件来检查文本文件。若在检查的文件中找到字典没有的词汇，ispell会建议使用的词汇，或是让你将新的词汇加入个人字典。

语法

```
ispell [-aAbBClmMnNPStVx][-d<字典文件>][-L<行数>][-p<字典文件>][-w<非字母字符>][-W<字符串长度>][
```

参数：

- -a 当其他程序输出送到ispell时，必须使用此参数。
- -A 读取到"&Include File&"字符串时，就去检查字符串后所指定文件的内容。
- -b 产生备份文件，文件名为.bak。
- -B 检查连字错误。
- -C 不检查连字错误。
- -d<字典文件> 指定字典文件。
- -l 从标准输入设备读取字符串，结束后显示拼错的词汇。
- -L<行数> 指定内文显示的行数。
- -m 自动考虑字尾的变化。
- -M 进入ispell后，在画面下方显示指令的按键。
- -n 检查的文件为noff或troff的格式。
- -N 进入ispell后，在画面下方不显示指令的按键。
- -p<字典文件> 指定个人字典文件。
- -P 不考虑字尾变化的情形。
- -S 不排序建议取代的词汇。
- -t 检查的文件为TeX或LaTeX的格式。
- -V 非ANSI标准的字符会以"M-^"的方式来显示。
- -w<非字母字符> 检查时，特别挑出含有指定的字符。
- -W<字符串长度> 不检查指定长度的词汇。
- -x 不要产生备份文件。

实例

检查文件的拼写。例如，检查testfile文件，可使用如下命令：

```
ispell testfile
```

如果文件中出现可疑词汇，则第一个出现的可疑词汇以高亮显示，并在屏幕下方给出词汇的修改意见，以及ispell的操作命令。如下所示：

```
netwrks File: testfile
Linux netwrks are becoming more and more common, but security is often an overlooked
issue. Unfortunately
0: networks
[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit e(X)it or ? for help
```

本例中，检查出netwrks 错误，并提示纠正信息，此时输入"0"，即使用networks 来纠正错误，同时继续显示下一个错误，直到所有的错误显示完毕。

通过以上实例我们可以发现，文件testfile中有拼写错误，对该文件进行修改后需备份文件。此时使用如下命令：

```
ispell-b testfile    #检查拼写错误的同时，备份文件
```

如果文件已经无拼写错误，则不显示任何信息，通过ls命令我们也可以查看到当前文件目录下产生了文件testfile的备份文件testfile.bak。查看结果如下所示：

```
$ ls #以列表的形式查看当前目录下的文件
examples.desktop testfile_1 testfile.bak xx01 模板图片 音乐
testfile testfile1 testfile_2 xx00 公共的视频文档桌面
```

其中，testfile.bak 文件就是刚才命令生成的备份文件，内容与原来的testfile 文件内容是一样的。

Linux jed命令

Linux jed命令用于编辑文本文件。

Jed是以Slang所写成的程序，适合用来编辑程序原始代码。

语法

```
jed [-2n][-batch][-f<函数>][-g<行数>][-i<文件>][-I<文件>][-s<字符串>][文件]
```

参数：

- -2 显示上下两个编辑区。
- -batch 以批处理模式来执行。
- -f<函数> 执行Slang函数。
- -g<行数> 移到缓冲区中指定的行数。
- -i<文件> 将指定的文件载入缓冲区。
- -i<文件> 载入Slang原始代码文件。
- -n 不要载入jed.rc配置文件。
- -s<字符串> 查找并移到指定的字符串。

实例

jed主要用于编辑程序的源码，编辑源码时将以彩色高亮的方式显示程序的语法。例如使用jed编辑一个C语言的源代码文件，可使用如下命令：

```
jed main.c       #用jed编辑器打开main.c 文件
```

输出结果如下：

```
F10 key ==> File Edit Mode Search Buffers Windows System Help #编辑器菜单
/*-*- linux-c-*-*/ #编辑区
#include <linux/mm.h>
#include <linux/sysctl.h>
#include <linux/nsproxy.h>
static struct list_head *
net_ctl_header_lookup(struct ctl_table_root *root, struct nsproxy *namespaces)
{
return &namespaces->net_ns->sysctl_table_headers;
}
static struct ctl_table_root net_sysctl_root = {
.lookup = net_ctl_header_lookup,
};
static int sysctl_net_init(struct net *net)
{
INIT_LIST_HEAD(&net->sysctl_table_headers);
return 0;
}
-----+(Jed 0.99.18U) Emacs: main.c (C) All 6:06pm-----------------------------
#从左到右分别为jed版本编号、当前是模拟emacs编辑器、打开的文件名、现在的时间
loading /usr/share/jed/lib/modeinfo.slc
```

Linux joe命令

Linux joe命令用于编辑文本文件。

Joe是一个功能强大的全屏幕文本编辑程序。操作的复杂度要比Pico高一点，但是功能较为齐全。Joe一次可开启多个文件，每个文件各放在一个编辑区内，并可在文件之间执行剪贴的动作。

语法

```
joe [-asis][-beep][-csmode][-dopadding][-exask][-force][-help][-keepup][-lightoff][-arkin
```

参数：

- 以下为程序参数 *-asis* 字符码超过*127*的字符不做任何处理。 -backpath<目录> 指定备份文件的目录。* -beep 编辑时，若有错误即发出哗声。
- -columns<栏位> 设置栏数。
- -csmode 可执行连续查找模式。
- -dopadding 是程序跟tty间存在缓冲区。
- -exask 在程序中，执行"Ctrl+k+x"时，会先确认是否要保存文件。
- -force 强制在最后一行的结尾处加上换行符号。
- -help 执行程序时一并显示帮助。
- -keepup 在进入程序后，画面上方为状态列。
- -lightoff 选取的区块在执行完区块命令后，就会回复成原来的状态。
- -lines<行数> 设置行数。
- -marking 在选取区块时，反白区块会随着光标移动。
- -mid 当光标移出画面时，即自动卷页，使光标回到中央。
- -nobackups 不建立备份文件。
- -nonotice 程序执行时，不显示版权信息。
- -nosta 程序执行时，不显示状态列。
- -noxon 尝试取消"Ctrl+s"与"Ctrl+q"键的功能。
- -orphan 若同时开启一个以上的文件，则其他文件会置于独立的缓冲区，而不会另外开启编辑区。
- -pg<行数> 按"PageUp"或"PageDown"换页时，所要保留前一页的行数。
- -skiptop<行数> 不使用屏幕上方指定的行数。
- 以下为文件参数
- +<行数> 指定开启文件时，光标所在的行数。

- -autoindent 自动缩排。
- -crlf 在换行时，使用CR-LF字符。
- -indentc<缩排字符> 执行缩排时，实际插入的字符。
- -istep<缩排字符数> 每次执行缩排时，所移动的缩排字符数。
- -keymap<按键配置文件> 使用不同的按键配置文件。
- -linums 在每行前面加上行号。
- -lmargin<栏数> 设置左侧边界。
- -overwrite 设置覆盖模式。
- -rmargin<栏数> 设置右侧边界。
- -tab<栏数> 设置tab的宽度。
- -rdonly 以只读的方式开启文件-wordwrap编辑时若超过右侧边界，则自动换行。

实例

利用joe命令编辑文本文件。例如利用joe编辑C 语言源代码main.c，使用如下命令：

```
joe main.c
```

与jed类似，joe编辑器中C语言的语法也以彩色的方式显示。效果如下：

```
I A main.c (c) Row 1 Col 1 12:28 Ctrl-K H for help
#上排从左至右分别为打开的文件名、光标所在行列数、现在时间、显示操作说明
/*-*- linux-c-*-*/ #编辑区
#include <linux/mm.h>
#include <linux/sysctl.h>
#include <linux/nsproxy.h>
static struct list_head *
net_ctl_header_lookup(struct ctl_table_root *root, struct nsproxy *namespaces)
{
return &namespaces->net_ns->sysctl_table_headers;
}
static struct ctl_table_root net_sysctl_root = {
.lookup = net_ctl_header_lookup,
};
static int sysctl_net_init(struct net *net)
{
INIT_LIST_HEAD(&net->sysctl_table_headers);
return 0;
}
** Joe's Own Editor v3.5 ** (utf-8) ** Copyright . 2006 ** #joe编辑区的版本及版权信息
```

joe编辑器有一些常用的组合键，例如可以通过Ctrl+K+H 寻求联机帮助，首先按Ctrl+K组合键，再输入字母H，即可调出帮助菜单，通过该帮助信息可以方便地获知如何对joe 编辑器进行操作。

Linux join命令

Linux join命令用于将两个文件中，指定栏位内容相同的行连接起来。

找出两个文件中，指定栏位内容相同的行，并加以合并，再输出到标准输出设备。

语法

```
join [-i][-a<1或2>][-e<字符串>][-o<格式>][-t<字符>][-v<1或2>][-1<栏位>][-2<栏位>][--help][--v
```

参数：

- -a<1或2> 除了显示原来的输出内容之外，还显示指令文件中没有相同栏位的行。
- -e<字符串> 若[文件1]与[文件2]中找不到指定的栏位，则在输出中填入选项中的字符串。
- -i或--igore-case 比较栏位内容时，忽略大小写的差异。
- -o<格式> 按照指定的格式来显示结果。
- -t<字符> 使用栏位的分隔字符。
- -v<1或2> 跟-a相同，但是只显示文件中没有相同栏位的行。
- -1<栏位> 连接[文件1]指定的栏位。
- -2<栏位> 连接[文件2]指定的栏位。
- --help 显示帮助。
- --version 显示版本信息。

实例

连接两个文件。

为了清楚地了解join命令，首先通过cat命令显示文件testfile_1和 testfile_2 的内容。

然后以默认的方式比较两个文件，将两个文件中指定字段的内容相同的行连接起来，在终端中输入命令：

```
join testfile_1 testfile_2
```

首先查看testfile_1、testfile_2 中的文件内容：

```
$ cat testfile_1 #testfile_1文件中的内容
Hello 95 #例如，本例中第一列为姓名，第二列为数额
Linux 85
test 30
cmd@hdd-desktop:~$ cat testfile_2 #testfile_2文件中的内容
Hello 2005 #例如，本例中第一列为姓名，第二列为年份
Linux 2009
test 2006
```

然后使用join命令，将两个文件连接，结果如下：

```
$ join testfile_1 testfile_2 #连接testfile_1、testfile_2中的内容
Hello 95 2005 #连接后显示的内容
Linux 85 2009
test 30 2006
```

文件1与文件2的位置对输出到标准输出的结果是有影响的。例如将命令中的两个文件互换，即输入如下命令：

```
join testfile_2 testfile_1
```

最终在标准输出的输出结果将发生变化，如下所示：

```
$ join testfile_2 testfile_1 #改变文件顺序连接两个文件
Hello 2005 95 #连接后显示的内容
Linux 2009 85
test 2006 30
```

Linux look命令

Linux look命令用于查询单词。

look指令用于英文单字的查询。您仅需给予它欲查询的字首字符串，它会显示所有开头字符串符合该条件的单字。

语法

```
look [-adf][-t<字尾字符串>][字首字符串][字典文件]
```

参数说明：

- -a 使用另一个字典文件web2，该文件也位于/usr/dict目录下。
- -d 只对比英文字母和数字，其余一概忽略不予比对。
- -f 忽略字符大小写差别。
- -t<字尾字符串> 设置字尾字符串。

实例

为了查找在testfile文件中以字母L开头的所有的行，可以输入如下命令：

```
look L testfile
```

原文件testfile中的内容如下：

```
$ cat testfile #查看testfile 文件内容
HELLO LINUX!
Linux is a free unix-type opterating system.
This is a linux testfile!
Linux test
```

在testfile文件中使用look命令查找以"L"开头的单词，结果如下：

```
$ look L testfile                              #查找以“L”开头的单词
Linux is a free unix-type opterating system.   #第二行以“L”开头，列出全句
Linux test                                     #第四行以“L”开头，列出全句
```

Linux mtype命令

mtype为mtools工具指令，模拟MS-DOS的type指令，可显示MS-DOS文件的内容。

语法

```
mtype [-st][文件]
```

参数说明：

- -s 去除8位字符码集的第一个位，使它兼容于7位的ASCII。
- -t 将MS-DOS文本文件中的"换行+光标移至行首"字符转换成Linux的换行字符。

实例

打开名为dos.txt 的MS-DOS文件可使用如下命令：

```
mtype dos.txt         #打开MS-DOS 文件
```

显示结果如下：

```
$ mtype dos.txt #打开MS-DOS 文件
Linux networks are becoming more and more common, but security is often an overlooked
issue. Unfortunately, in today’s environment all networks are potential hacker targets,
from top-secret military research networks to small home LANs.
Linux Network Securty focuses on securing Linux in a networked environment, where the
security of the entire network needs to be considered rather than just isolated machines.
It uses a mix of theory and practicl techniques to teach administrators how to install an
use security applications, as well as how the applcations work and why they are necessary
```

Linux pico命令

Linux pico命令用于编辑文字文件。

pico是个简单易用、以显示导向为主的文字编辑程序，它伴随着处理电子邮件和新闻组的程序pine而来。

语法

```
pico [-bdefghjkmqtvwxz][-n<间隔秒数>][-o<工作目录>][-r<编辑页宽>][-s<拼字检查器>][+<列数编号>][文
```

参数说明：

- -b 开启置换的功能。
- -d 开启删除的功能。
- -e 使用完整的文件名称。
- -f 支持键盘上的F1、F2...等功能键。
- -g 显示光标。
- -h 在线帮助。
- -j 开启切换的功能。
- -k 预设pico在使用剪下命令时，会把光标所在的列的内容全部删除。
- -m 开启鼠标支持的功能，您可用鼠标点选命令列表。
- -n<间隔秒数> 设置多久检查一次新邮件。
- -o<工作目录> 设置工作目录。
- -q 忽略预设值。
- -r<编辑页宽> 设置编辑文件的页宽。
- -s<拼字检查器> 另外指定拼字检查器。
- -t 启动工具模式。
- -v 启动阅读模式，用户只能观看，无法编辑文件的内容。
- -w 关闭自动换行，通过这个参数可以编辑内容很长的列。
- -x 关闭换面下方的命令列表。
- -z 让pico可被Ctrl+z中断，暂存在后台作业里。
- +<列数编号> 执行pico指令进入编辑模式时，从指定的列数开始编辑。

实例

使用pico命令来编辑testfile文件，在终端中输入如下命令：

```
pico testfile
```

输出结果如下：

```
GNU nano 2.0.9 文件: testfile #从左到右分别为编辑器版本号、文件名
#编辑区
Linux networks are becoming more and more common, but security is often an over$
Linux Network Securty focuses on securing Linux in a networked environment, whe$
[ 已读取3 行] #以下为菜单栏
^G 求助^O 写入^R 读档^Y 上页^K 剪切文字^C 在标位置
^X 离开^J 对齐^W 搜寻^V 下页^U 还原剪切^T 拼写检查
```

Linux rgrep命令

Linux rgrep命令用于递归查找文件里符合条件的字符串。

rgrep指令的功能和grep指令类似，可查找内容包含指定的范本样式的文件，如果发现某文件的内容符合所指定的范本样式，预设rgrep指令会把含有范本样式的那一列显示出来。

语法

```
rgrep [-?BcDFhHilnNrv][-R<范本样式>][-W<列长度>][-x<扩展名>][--help][--version][范本样式][文件
```

参说明数：

- -? 显示范本样式与范例的说明。
- -B 忽略二进制的数据。
- -c 计算符合范本样式的列数。
- -D 排错模式，只列出指令搜寻的目录清单，而不会读取文件内容。
- -F 当遇到符号连接时，rgrep预设是忽略不予处理，加上本参数后，rgrep指令就会读取该连接所指向的原始文件的内容。
- -h 特别将符合范本样式的字符串标示出来。
- -H 只列出符合范本样式的字符串，而非显示整列的内容。
- -i 忽略字符大小写的差别。
- -l 列出文件内容符合指定的范本样式的文件名称。
- -n 在显示符合坊本样式的那一列之前，标示出该列的列数编号。
- -N 不要递归处理。
- -r 递归处理，将指定目录下的所有文件及子目录一并处理。
- -R<范本样式> 此参数的效果和指定"-r"参数类似，但只主力符合范本样式文件名称的文件。
- -v 反转查找。
- -W<列长度> 限制符合范本样式的字符串所在列，必须拥有的字符数。
- -x<扩展名> 只处理符合指定扩展名的文件名称的文件。
- --help 在线帮助。
- --version 显示版本信息。

实例

在当前目录下查找句子中包含"Hello"字符串的文件，可使用如下命令：

```
rgrep Hello *
```

其搜索结果如下：

```
$ rgrep Hello *             #在当前目录下查找句子中包含“Hello”字符串的文件
testfile_1:Hello 95         #testfile_1中包含“Hello”字符串的句子
testfile_2:Hello 2005       #testfile_2中包含“Hello”字符串的句子
```

Linux sed命令

Linux sed命令是利用script来处理文本文件。

sed可依照script的指令，来处理、编辑文本文件。

语法

```
sed [-hnV][-e<script>][-f<script文件>][文本文件]
```

参数说明：

- -e<script>或--expression=<script> 以选项中指定的script来处理输入的文本文件。
- -f<script文件>或--file=<script文件> 以选项中指定的script文件来处理输入的文本文件。
- -h或--help 显示帮助。
- -n或--quiet或--silent 仅显示script处理后的结果。
- -V或--version 显示版本信息。

实例

在testfile文件的第四行后添加一行，并将结果输出到标准输出，在命令行提示符下输入如下命令：

```
sed -e 4a\newLine testfile
```

首先查看testfile中的内容如下：

```
$ cat testfile #查看testfile 中的内容
HELLO LINUX!
Linux is a free unix-type opterating system.
This is a linux testfile!
Linux test
```

使用sed命令后，输出结果如下：

```
$ sed -e 4a\newline testfile #使用sed 在第四行后添加新字符串
HELLO LINUX! #testfile文件原有的内容
Linux is a free unix-type opterating system.
This is a linux testfile!
Linux test
newline
```

Linux sort命令

Linux sort命令用于将文本文件内容加以排序。

sort可针对文本文件的内容，以行为单位来排序。

语法

```
sort [-bcdfimMnr][-o<输出文件>][-t<分隔字符>][+<起始栏位>-<结束栏位>][--help][--verison][文件]
```

参数说明：

- -b 忽略每行前面开始出的空格字符。
- -c 检查文件是否已经按照顺序排序。
- -d 排序时，处理英文字母、数字及空格字符外，忽略其他的字符。
- -f 排序时，将小写字母视为大写字母。
- -i 排序时，除了040至176之间的ASCII字符外，忽略其他的字符。
- -m 将几个排序好的文件进行合并。
- -M 将前面3个字母依照月份的缩写进行排序。
- -n 依照数值的大小排序。
- -o<输出文件> 将排序后的结果存入指定的文件。
- -r 以相反的顺序来排序。
- -t<分隔字符> 指定排序时所用的栏位分隔字符。
- +<起始栏位>-<结束栏位> 以指定的栏位来排序，范围由起始栏位到结束栏位的前一栏位。
- --help 显示帮助。
- --version 显示版本信息。

实例

在使用sort命令以默认的式对文件的行进行排序，使用的命令如下：

```
sort testfile
```

sort 命令将以默认的方式将文本文件的第一列以ASCII 码的次序排列，并将结果输出到标准输出。

使用 cat命令显示testfile文件可知其原有的排序如下：

```
$ cat testfile      #testfile文件原有排序
test 30
Hello 95
Linux 85
```

使用sort命令重排后的结果如下：

```
$ sort testfile #重排结果
Hello 95
Linux 85
test 30
```

Linux spell命令

Linux spell命令可建立拼写检查程序。

spell可从标准输入设备读取字符串，结束后显示拼错的词汇。

语法

```
spell
```

实例

检查文件testfile是否有拼写错误，在命令行提示符下输入如下命令：

```
spell testfile
```

如果文件中有单词拼写错误，则输出如下信息：

```
$ spell testfile   #检查testfile 拼写错误
scurity            #以下为有错误的单词
tp
LANs
Securty
practicl
applcations
necesary
```

如果所检查的文件没有单词拼写错误，那么，命令运行后不会给出任何信息。

检查从标准输入读取的字符串。例如在命令行中输入如下命令：

```
spell
```

按回车键后，输入一串字符串，然后按Ctrl+D 组合键退出spell，屏幕上将显示拼写有错误的单词。如下所示：

```
$ spell #检查标准输入的字符串的拼写错误
hell,this is a linx sustem! #拼写错误的字符串
linx #以下为有拼写错误的单词
sustem
```

Linux tr命令

Linux tr 命令用于转换或删除文件中的字符。

tr 指令从标准输入设备读取数据，经过字符串转译后，将结果输出到标准输出设备。

语法

```
tr [-cdst][--help][--version][第一字符集][第二字符集]
tr [OPTION]…SET1[SET2]
```

参数说明：

- -c, --complement：反选设定字符。也就是符合 SET1 的部份不做处理，不符合的剩余部份才进行转换
- -d, --delete：删除指令字符
- -s, --squeeze-repeats：缩减连续重复的字符成指定的单个字符
- -t, --truncate-set1：削减 SET1 指定范围，使之与 SET2 设定长度相等
- --help：显示程序用法信息
- --version：显示程序本身的版本信息

字符集合的范围：

- \NNN 八进制值的字符 NNN (1 to 3 为八进制值的字符)
- \ 反斜杠
- \a Ctrl-G 铃声
- \b Ctrl-H 退格符
- \f Ctrl-L 走行换页
- \n Ctrl-J 新行
- \r Ctrl-M 回车
- \t Ctrl-I tab键
- \v Ctrl-X 水平制表符
- CHAR1-CHAR2 ：字符范围从 CHAR1 到 CHAR2 的指定，范围的指定以 ASCII 码的次序为基础，只能由小到大，不能由大到小。
- [CHAR*] ：这是 SET2 专用的设定，功能是重复指定的字符到与 SET1 相同长度为止
- [CHAR*REPEAT] ：这也是 SET2 专用的设定，功能是重复指定的字符到设定的 REPEAT 次数为止(REPEAT 的数字采 8 进位制计算，以 0 为开始)
- [:alnum:] ：所有字母字符与数字
- [:alpha:] ：所有字母字符
- [:blank:] ：所有水平空格

- [:cntrl:] ：所有控制字符
- [:digit:] ：所有数字
- [:graph:] ：所有可打印的字符(不包含空格符)
- [:lower:] ：所有小写字母
- [:print:] ：所有可打印的字符(包含空格符)
- [:punct:] ：所有标点字符
- [:space:] ：所有水平与垂直空格符
- [:upper:] ：所有大写字母
- [:xdigit:] ：所有 16 进位制的数字
- [=CHAR=] ：所有符合指定的字符(等号里的 CHAR，代表你可自订的字符)

实例

将文件testfile中的小写字母全部转换成大写字母，此时，可使用如下命令：

```
cat testfile |tr a-z A-Z
```

testfile文件中的内容如下：

```
$ cat testfile         #testfile原来的内容
Linux networks are becoming more and more common,
but scurity is often an overlooked
issue. Unfortunately, in today’s environment all networks
are potential hacker targets,
fro0m tp-secret military research networks to small home LANs.
Linux Network Securty focuses on securing Linux in a
networked environment, where the
security of the entire network needs to be considered
rather than just isolated machines.
It uses a mix of theory and practicl techniques to
teach administrators how to install and
use security applications, as well as how the
applcations work and why they are necesary.
```

使用 tr 命令大小写转换后，得到如下输出结果：

```
$ cat testfile | tr a-z A-Z #转换后的输出
LINUX NETWORKS ARE BECOMING MORE AND MORE COMMON, BUT SCURITY IS OFTEN AN OVERLOOKED
ISSUE. UNFORTUNATELY, IN TODAY’S ENVIRONMENT ALL NETWORKS ARE POTENTIAL HACKER TARGETS,
FROM TP-SECRET MILITARY RESEARCH NETWORKS TO SMALL HOME LANS.
LINUX NETWORK SECURTY FOCUSES ON SECURING LINUX IN A NETWORKED ENVIRONMENT, WHERE THE
SECURITY OF THE ENTIRE NETWORK NEEDS TO BE CONSIDERED RATHER THAN JUST ISOLATED MACHINES.
IT USES A MIX OF THEORY AND PRACTICL TECHNIQUES TO TEACH ADMINISTRATORS HOW TO INSTALL AN
USE SECURITY APPLICATIONS, AS WELL AS HOW THE APPLCATIONS WORK AND WHY THEY ARE NECESARY.
```

大小写转换，也可以通过[:lower][:upper]参数来实现。例如使用如下命令：

```
cat testfile |tr [:lower:] [:upper:]
```

输出结果如下：

```
$ cat testfile | tr [:lower:] [:upper:] #转换后的输出
LINUX NETWORKS ARE BECOMING MORE AND MORE COMMON, BUT SCURITY IS OFTEN AN OVERLOOKED
ISSUE. UNFORTUNATELY, IN TODAY’S ENVIRONMENT ALL NETWORKS ARE POTENTIAL HACKER TARGETS,
FROM TP-SECRET MILITARY RESEARCH NETWORKS TO SMALL HOME LANS.
LINUX NETWORK SECURTY FOCUSES ON SECURING LINUX IN A NETWORKED ENVIRONMENT, WHERE THE
SECURITY OF THE ENTIRE NETWORK NEEDS TO BE CONSIDERED RATHER THAN JUST ISOLATED MACHINES.
IT USES A MIX OF THEORY AND PRACTICL TECHNIQUES TO TEACH ADMINISTRATORS HOW TO INSTALL AN
USE SECURITY APPLICATIONS, AS WELL AS HOW THE APPLCATIONS WORK AND WHY THEY ARE NECESARY.
```

Linux expr命令

expr命令是一个手工命令行计数器，用于在UNIX/LINUX下求表达式变量的值，一般用于整数值，也可用于字符串。

语法

```
expr 表达式
```

表达式说明:

- 用空格隔开每个项；
- 用 / (反斜杠) 放在 shell 特定的字符前面；
- 对包含空格和其他特殊字符的字符串要用引号括起来

实例

1、计算字串长度

```
> expr length “this is a test”
 14
```

2、抓取字串

```
> expr substr “this is a test” 3 5
is is
```

3、抓取第一个字符数字串出现的位置

```
> expr index "sarasara"  a
 2
```

4、整数运算

```
> expr 14 % 9
5
> expr 10 + 10
20
> expr 1000 + 900
1900
> expr 30 / 3 / 2
5
> expr 30 /* 3 (使用乘号时，必须用反斜线屏蔽其特定含义。因为shell可能会误解显示星号的意义)
90
> expr 30 * 3
expr: Syntax error
```

Linux uniq命令

Linux uniq命令用于检查及删除文本文件中重复出现的行列。

uniq可检查文本文件中重复出现的行列。

语法

```
uniq [-cdu][-f<栏位>][-s<字符位置>][-w<字符位置>][--help][--version][输入文件][输出文件]
```

参数：

- -c或--count 在每列旁边显示该行重复出现的次数。
- -d或--repeated 仅显示重复出现的行列。
- -f<栏位>或--skip-fields=<栏位> 忽略比较指定的栏位。
- -s<字符位置>或--skip-chars=<字符位置> 忽略比较指定的字符。
- -u或--unique 仅显示出一次的行列。
- -w<字符位置>或--check-chars=<字符位置> 指定要比较的字符。
- --help 显示帮助。
- --version 显示版本信息。
- [输入文件] 指定已排序好的文本文件。
- [输出文件] 指定输出的文件。

实例

文件testfile中第2 行、第5 行、第9 行为相同的行，使用uniq 命令删除重复的行，可使用以下命令：

```
uniq testfile
```

testfile中的原有内容为：

```
$ cat testfile      #原有内容
test 30
test 30
test 30
Hello 95
Hello 95
Hello 95
Hello 95
Linux 85
Linux 85
```

使用uniq 命令删除重复的行后，有如下输出结果：

```
$ uniq testfile     #删除重复行后的内容
test 30
Hello 95
Linux 85
```

检查文件并删除文件中重复出现的行，并在行首显示该行重复出现的次数。使用如下命令：

```
uniq-c testfile
```

结果输出如下：

```
$ uniq-ctestfile      #删除重复行后的内容
3 test 30             #前面的数字的意义为该行共出现了3次
4 Hello 95            #前面的数字的意义为该行共出现了4次
2 Linux 85            #前面的数字的意义为该行共出现了2次
```

Linux wc命令

Linux wc命令用于计算字数。

利用wc指令我们可以计算文件的Byte数、字数、或是列数，若不指定文件名称、或是所给予的文件名为"-"，则wc指令会从标准输入设备读取数据。

语法

```
wc [-clw][--help][--version][文件...]
```

参数：

- -c或--bytes或--chars 只显示Bytes数。
- -l或--lines 只显示列数。
- -w或--words 只显示字数。
- --help 在线帮助。
- --version 显示版本信息。

实例

在默认的情况下，wc将计算指定文件的行数、字数，以及字节数。使用的命令为：

```
wc testfile
```

先查看testfile文件的内容，可以看到：

```
$ cat testfile
Linux networks are becoming more and more common, but scurity is often an overlooked
issue. Unfortunately, in today’s environment all networks are potential hacker targets,
fro0m tp-secret military research networks to small home LANs.
Linux Network Securty focuses on securing Linux in a networked environment, where the
security of the entire network needs to be considered rather than just isolated machines.
It uses a mix of theory and practicl techniques to teach administrators how to install an
use security applications, as well as how the applcations work and why they are necesary.
```

使用 wc统计，结果如下：

```
$ wc testfile           # testfile文件的统计信息
3 92 598 testfile       # testfile文件的行数为3、单词数92、字节数598
```

其中，3 个数字分别表示testfile文件的行数、单词数，以及该文件的字节数。

如果想同时统计多个文件的信息，例如同时统计testfile、testfile_1、testfile_2，可使用如下命令：

```
wc testfile testfile_1 testfile_2   #统计三个文件的信息
```

输出结果如下：

```
$ wc testfile testfile_1 testfile_2  #统计三个文件的信息
3 92 598 testfile                    #第一个文件行数为3、单词数92、字节数598
9 18 78 testfile_1                   #第二个文件的行数为9、单词数18、字节数78
3 6 32 testfile_2                    #第三个文件的行数为3、单词数6、字节数32
15 116 708 总用量                    #三个文件总共的行数为15、单词数116、字节数708
```

Linux命令大全 - 文件传输

lprm	lpr	lpq	lpd
bye	ftp	uuto	uupick
uucp	uucico	tftp	ncftp
ftpshut	ftpwho	ftpcount	

Linux lprm命令

Linux lprm命令用于将一个工作由打印机贮列中移除

尚未完成的打印机工作会被放在打印机贮列之中，这个命令可用来将常未送到打印机的工作取消。由于每一个打印机都有一个独立的贮列，你可以用 -P 这个命令设定想要作用的印列机。如果没有设定的话，会使用系统预设的打印机。

这个命令会检查使用者是否有足够的权限删除指定的档案，一般而言，只有档案的拥有者或是系统管理员才有这个权限。

语法

```
/usr/bin/lprm [P] [file...]
```

实例

将打印机 hpprinter 中的第 1123 号工作移除

```
lprm -Phpprinter 1123
```

将第 1011 号工作由预设印表机中移除

```
lprm 1011
```

Linux lpr命令

lpr(line printer，按行打印)实用程序用来将一个或多个文件放入打印队列等待打印。

lpr 可以用来将料资送给本地或是远端的主机来处理。

语法

```
lpr [ -P printer ]
```

参数：

- -p Printer: 将资料送至指定的打印机 Printer，预设值为 lp。

实例

下面的命令行将在名为mailroom的打印机上打印report文件：

```
$ lpr -P mailroom report
```

使用一条打印命令可打印多个文件，下面的命令行在名为laser1的打印机上打印3个文件：

```
$ lpr -P laser1 05.txt 108.txt 12.txt
```

Linux lpq命令

Linux lpq命令用于查看一个打印队列的状态，该程序可以查看打印机队列状态及其所包含的打印任务。

语法

lpq [l] [P] [user]

参数说明：

- -P 指定一个打印机，否则使用默认打印机或环境变量PRINTER指定的打印机
- -l 打印组成作业的所有文件的信息。。

实例

为系统默认的打印机printer的一个空队列。

```
$ lpq
printer is ready
no entries
```

如果事先并未指定打印机（使用-P选项），系统便会显示默认的打印机。如果向打印机发送打印任务，然后查看打印队列，便会看到如下列表。

```
$ ls *.txt | pr -3 | lp
request id is printer-603 (1 file(s))
[me@linuxbox ~]$ lpq
printer is ready and printing
Rank     Owner    Job     File(s)                         Total Size
active   me       603     (stdin)
```

Linux lpd命令

Linux lpd命令 是一个常驻的打印机管理程序，它会根据 /etc/printcap 的内容来管理本地或远端的打印机。

/etc/printcap 中定义的每一个打印机必须在 /var/lpd 中有一个相对应的目录，目录中以 cf 开头的档案表示一个等待送到适当装置的印表工作。这个档案通常是由 lpr 所产生。

lpr 和 lpd 组成了一个可以离线工作的系统，当你使用 lpr 时，打印机不需要能立即可用，甚至不用存在。

lpd 会自动监视打印机的状况，当打印机上线后，便立即将档案送交处理。这个得所有的应用程序不必等待打印机完成前一工作。

语法

```
lpd [-l] [#port]
```

参数说明：

- -l: 将一些除错讯息显示在标准输出上。

- # port: 一般而言，lpd 会使用 getservbyname 取得适当的 TCP/IP port，你可以使用这个参数强迫 lpd 使用指定的 port。

实例

这个程序通常是由 /etc/rc.d 中的程序在系统启始阶段执行。

Linux bye命令

Linux bye命令用于中断FTP连线并结束程序。

在ftp模式下，输入bye即可中断目前的连线作业，并结束ftp的执行。

语法

```
bye
```

Linux ftp命令

Linux ftp命令设置文件系统相关功能。

FTP是ARPANet的标准文件传输协议，该网络就是现今Internet的前身。

语法

```
ftp [-dignv][主机名称或IP地址]
```

参数：

- -d 详细显示指令执行过程，便于排错或分析程序执行的情形。
- -i 关闭互动模式，不询问任何问题。
- -g 关闭本地主机文件名称支持特殊字符的扩充特性。
- -n 不使用自动登陆。
- -v 显示指令执行过程。

实例

例如使用ftp命令匿名登录ftp.kernel.org服务器，该服务是Linux 内核的官方服务器，可以使用如下命令：

```
ftp ftp.kernel.org #发起链接请求
```

Linux ncftp命令

Linux ncftp命令用于传输文件。

FTP让用户得以下载存放于服务器主机的文件，也能将文件上传到远端主机放置。

NcFTP是文字模式FTP程序的佼佼者，它具备多样特色， 包括显示传输速率，下载进度，自动续传，标住书签，可通过防火墙和代理服务器等。

当不指定用户名时，ncftp 命令会自动尝试使用匿名账户anonymous 去连接远程FTP 服 务器，不需要用户输入账号和密码。

语法

```
ncftp [主机或IP地址]
```

参数说明：

- -u<用户名> 指定登录FTP服务器的用户名
- -p<密码> 设置用户密码
- -P<端口号> 指定FTP端口号，默认为21
- -j<账号> 指定账号
- -h 帮助信息
- -v 版本信息

实例

使用ncftp命令匿名连接FTP服务器。

例如想匿名连接ftp.kernel.org服务器，同时不想输入anonymous等匿名用户名，可直接使用ncftp命令：

```
ncftp ftp.kernel.org
```

得到如下信息：

```
$ ncftp ftp.kernel.org #匿名连接ftp.kernel.org服务器
NcFTP 3.2.1 (Jul 29, 2007) by Mike Gleason (http://www.NcFTP.com/contact/).
#ncftp版权、版本等信息
Copyright (c) 1992-2005 by Mike Gleason.
All rights reserved.
Connecting to 149.20.20.133... #连接服务器
Welcome to ftp.kernel.org.
Logging in... #匿名登录
Welcome to the #欢迎信息
LINUX KERNEL ARCHIVES
ftp.kernel.org
"Much more than just kernels"
IF YOU'RE ACCESSING THIS SITE VIA A WEB BROWSER
PLEASE USE THE HTTP URL BELOW INSTEAD!
----> If you are looking for mirror sites, please go <----
----> to mirrors.kernel.org instead <----
This site is provided as a public service by the Linux Kernel
Organization, a California nonprofit corporation. Bandwidth is
provided by The Internet Software Consortium, Inc. Our servers are
located in San Francisco and Palo Alto, California; Corvallis, Oregon;
Amsterdam, Netherlands and Ume., Sweden; use in violation of any
applicable laws strictly prohibited.
Due to U.S. Exports Regulations, all cryptographic software on this
site is subject to the following legal notice:
This site includes publicly available encryption source code
which, together with object code resulting from the compiling of
publicly available source code, may be exported from the United
States under License Exception "TSU" pursuant to 15 C.F.R. Section
740.13(e).
This legal notice applies to cryptographic software only. Please see
the Bureau of Industry and Security (http://www.bis.doc.gov/) for more
information about current U.S. regulations.
Neither the Linux Kernel Organization, nor its sponsors make any
guarantees, explicit or implicit, about the contents of this site.
Use at your own risk.
This site is accessible via the following mechanisms:
FTP ftp://ftp.kernel.org/pub/
HTTP http://www.kernel.org/pub/
RSYNC rsync://rsync.kernel.org/pub/
NFS and SMB/CIFS are no longer available.
For comments on this site, please contact <ftpadmin@kernel.org>.
Please do not use this address for questions that are not related to
the operation of this site. Please see our homepage at
http://www.kernel.org/ for links to Linux documentation resources.
Login successful.
Logged in to ftp.kernel.org.
ncftp / >
```

提示：ncftp的命令提示符为"ncftp / >"，而不是ftp中的"ftp / >"。

使用ncftp命令操作、下载文件。

ncftp的命令基本上与ftp相同，例如可以使用"cd"命令切换在FTP服务器中的当前目录，使用"ls"命令列出当前目录内容，使用"get"命令下载"/pub"目录下的README文件、使用"quit"离开ncftp等。操作结果如下：

```
ncftp / > pwd                #查看当前路径
ftp://ftp.kernel.org         #当前路径为根目录
ncftp / > ls                 #查看当前目录列表
bin/ for_mirrors_only/ pub/
dev/ lib/ usr@
etc/ lost+found/ welcome.msg@
ncftp / > cd pub             #切换目录到pub 子目录
Directory successfully changed.
ncftp /pub > ls              #查看pub 的目录列表
dist/ media/ scm/
index.html RCS/ site/
linux/ README software/
lost+found/ README_ABOUT_BZ2_FILES tools/
ncftp /pub > get README      #下载README 文件
README: 1.87 KB 10.39 KB/s
ncftp /pub > quit            #离开ncftp
```

与ftp不同的是，ncftp此时会提示用户是否将FTP服务器保存为书签，以便于下次登录，用户可以进行自定义书签名等操作，如下所示：

```
You have not saved a bookmark for this site.  #离开提示信息
Would you like to save a bookmark to:
ftp://ftp.kernel.org/pub/
Save? (yes/no) yes                            #确认是否保存
Enter a name for this bookmark, or hit enter for "kernel": kernel #输入书签名
Bookmark "kernel" saved.
```

Linux tftp命令

Linux tftp命令用于传输文件。

FTP让用户得以下载存放于远端主机的文件，也能将文件上传到远端主机放置。tftp是简单的文字模式ftp程序，它所使用的指令和FTP类似。

语法

```
tftp [主机名称或IP地址]
```

操作说明：

- connect：连接到远程tftp服务器
- mode：文件传输模式
- put：上传文件
- get：下载文件
- quit：退出
- verbose：显示详细的处理信息
- tarce：显示包路径
- status：显示当前状态信息
- binary：二进制传输模式
- ascii：ascii 传送模式
- rexmt：设置包传输的超时时间
- timeout：设置重传的超时时间
- help：帮助信息
- ？：帮助信息

实例

连接远程服务器"218.28.188.288"，然后使用put 命令下载其中根目录下的文件"README"，可使用命令如下：

```
tftp 218.28.188.288 #连接远程服务器
```

连接服务器之后可进行相应的操作，具体如下：

```
$ tftp 218.28.188.228                           #连接远程服务器
tftp> ?                                         #使用？，参考帮助
Commands may be abbreviated. Commands are: #帮助命令列表
connect connect to remote tftp
mode set file transfer mode
put send file
get receive file
quit exit tftp
verbose toggle verbose mode
trace toggle packet tracing
status show current status
binary set mode to octet
ascii set mode to netascii
rexmt set per-packet retransmission timeout
timeout set total retransmission timeout
? print help information
tftp>get README                                  #远程下载README文件
getting from 218.28.188.288 to /home/cmd
Recived 168236 bytes in 1.5 seconds[112157 bit/s]
tftp>quit                                        #离开tftp
```

Linux uuto命令

Linux uuto命令将文件传送到远端的UUCP主机。

uuto为script文件，它实际上会执行uucp，用来将文件传送到远端UUCP主机，并在完成工作后，以邮件通知远端主机上的用户。

语法

```
uuto [文件][目的]
```

参数：

相关参数请参考 uucp指令。

实例

将文件传送到远程UUCP主机localhost的tmp 目录，在命令提示符中直接输入如下命令：

```
uuto./testfile localhost/tmp #将文件传送到远程UUCP 主机localhost的tmp目录
```

该命令通常没有输出。

Linux uupick命令

Linux uupick命令处理传送进来的文件。

当其他主机通过UUCP将文件传送进来时，可利用uupick指令取出这些文件。

语法

```
uupick [-v][-I<配置文件>][-s<主机>][-x<层级>][--help]
```

参数：

- -I<配置文件>或--config<配置文件> 指定配置文件。
- -s<主机>或--system<主机> 处理由指定主机传送过来的文件。
- -v或--version 显示版本信息。
- --help 显示帮助。

实例

处理由主机localhost传送过来的文件。在命令行直接输入如下命令：

```
uupick-s localhost
```

该命令通常没有输出。

Linux uucp命令

Linux uucp命令用于在Unix系统之间传送文件。

UUCP为Unix系统之间，通过序列线来连线的协议。uucp使用UUCP协议，主要的功能为传送文件。

语法

```
uucp [-cCdfjmrRtvW][-g<等级>][-I<配置文件>][-n<用户>][-x<类型>][--help][...来源][目的]
```

参数说明：

- -c或--nocopy 不用将文件复制到缓冲区。
- -C或--copy 将文件复制到缓冲区。
- -d或--directiories 在传送文件时，自动在[目的]建立必要的目录。
- -f或--nodirectiories 在传送文件时，若需要在[目的]建立目录，则放弃执行该作业。
- -g<等级>或--grade<等级> 指定文件传送作业的优先顺序。
- -I<配置文件>或--config<配置文件> 指定uucp配置文件。
- -j或--jobid 显示作业编号。
- -m或--mail 作业结束后，以电子邮件报告作业是否顺利完成。
- -n<用户>或--notify<用户> 作业结束后，以电子邮件向指定的用户报告作业是否顺利完成。
- -r或--nouucico 不要立即启动uucico服务程序，仅将作业送到队列中，待稍后再执行。
- -R或--recursive 若[来源]为目录，则将整个目录包含子目录复制到[目的]。
- -t或--uuto 将最后一个参数视为"主机名!用户"。
- -v或--version 显示版本信息。
- -W或--noexpand 不要将目前所在的目录加入路径。
- -x<类型>或--debug<类型>启动指定的排错模式。
- --help 显示帮助。
- [源...] 指定源文件或路径。
- [目的] 指定目标文件或路径。

实例

将temp/目录下所有文件传送到远程主机localhost的uucp公共目录下的Public/目录下。在命令行中输入如下命令：

```
uucp-d-R temp localhost ~/Public/
```

该命令通常没有输出

Linux uucico命令

Linux uucico命令UUCP文件传输服务程序。

uucico是用来处理uucp或uux送到队列的文件传输工具。uucico有两种工作模式：主动模式和附属模式。当在主动模式下时，uucico会调用远端主机；在附属模式下时，uucico则接受远端主机的调用。

语法

```
uucico [-cCDefqvwz][-i<类型>][-I<文件>][-p<连接端口号码>][-][-rl][-s<主机>][-S<主机>][-u<用户>
```

参数说明

- -c或--quiet 当不执行任何工作时，不要更改记录文件的内容及更新目前的状态。
- -C或--ifwork 当有工作要执行时，才调用-s或-S参数所指定主机。
- -D或--nodetach 不要与控制终端机离线。
- -e或--loop 在附属模式下执行，并且出现要求登入的提示画面。
- -f或--force 当执行错误时，不等待任何时间即重新调用主机。
- -i<类型>或--stdin<类型> 当使用到标准输入设备时，指定连接端口的类型。
- -I<文件>--config<文件> 指定使用的配置文件。
- -l或--prompt 出现要求登入的提示画面。
- -p<连接端口号码>或-port<连接端口号码> 指定连接端口号码。
- -q或--quiet 不要启动uuxqt服务程序。
- -r0或--slave 以附属模式启动。
- -s<主机>或--system<主机> 调用指定的主机。
- -u<用户>或--login<用户> 指定登入的用户帐号，而不允许输入任意的登入帐号。
- -v或--version 显示版本信息，并且结束程序。
- -w或--wait 在主动模式下，当执行调用动作时，则出现要求登入的提示画面。
- -x<类型>或-X<类型>或outgoing-debug<类型> 启动指定的排错模式。
- -z或--try-next 当执行不成功时，尝试下一个选择而不结束程序。
- --help 显示帮助，并且结束程序。

实例

使用主动模式启动uucico服务。在命令提示符下直接输入如下命令：

```
uucico-r1
```

提示：该命令一般没有输出。

Linux ftpshut命令

Linux ftpshut命令在指定的时间关闭FTP服务器。

本指令提供系统管理者在设置的时间关闭FTP服务器，且能在关闭之前发出警告信息通知用户。关闭时间若设置后为"none"，则会马上关闭服务器。如果采用"+30"的方式来设置表示服务器在30分钟之后关闭。依次类推，假设使用"1130"的格式则代表服务器会在每日的11时30分关闭，时间格式为24小时制。FTP服务器关闭后，在/etc目录下会产生一个名称为shutmsg的文件，把它删除后即可再度启动FTP服务器的功能。

语法

```
ftpshut [-d<分钟>][-l<分钟>][关闭时间]["警告信息"]
```

参数：

- -d<分钟> 切断所有FTP连线时间。
- -l<分钟> 停止接受FTP登入的时间。

实例

在晚上11:00 关闭FTP服务器，并在关闭前5 分钟拒绝新的FTP登录，前3 分钟关闭所有ftp的链接，且给出警告信息，可使用如下命令：

```
ftpshut-d 3 -1 5 1100 "Server will be shutdown at 23:00:00"
```

Linux ftpwho命令

Linux ftpwho命令用于显示目前所有以FTP登入的用户信息。

执行这项指令可得知目前用FTP登入系统的用户有那些人，以及他们正在进行的操作。

语法

```
ftpwho
```

参数说明：

- -v 显示版本信息

实例

查询当前有哪些用户正在登录FTP服务器，可直接使用如下命令：

```
ftpwho
```

该命令有如下输出结果：

```
$ ftpwho              #查询当前正在登录FTP 服务器的用户
standalone FTP daemon[2085]：
3547 wyw [1m20s] 1m25s(idle)
Service class - 1 user #当前有一个用户登录FTP服务器
```

Linux ftpcount命令

Linux ftpcount命令用于显示目前以FTP登入的用户人数。

执行这项指令可得知目前用FTP登入系统的人数以及FTP登入人数的上限。

语法

```
ftpcount
```

参数说明：

- -f<设定文件>：指定设定文件的路径。
- -h，--help ：显示帮助信息。

实例

ftpcount 可以直接查询FTP服务器上用户的人数，可直接使用如下命令：

```
ftpcount              #查询当前FTP用户的人数
```

该命令有如下输出结果：

```
$ ftpcount              #查询当前FTP用户的人数
Master proftpd process 2085:
Service class - 6 user #当前共6个用户登录到服务器
```

Linux命令大全 - 磁盘管理

cd	df	dirs	du
edquota	eject	mcd	mdeltree
mdu	mkdir	mlabel	mmd
mrd	mzip	pwd	quota
mount	mmount	rmdir	rmt
stat	tree	umount	ls
quotacheck	quotaoff	lndir	repquota
quotaon			

Linux cd命令

Linux cd命令用于切换当前工作目录至 dirName(目录参数)。

其中 dirName 表示法可为绝对路径或相对路径。若目录名称省略，则变换至使用者的 home 目录 (也就是刚 login 时所在的目录)。

另外，"~" 也表示为 home 目录 的意思，"." 则是表示目前所在的目录，".." 则表示目前目录位置的上一层目录。

语法

```
cd [dirName]
```

- dirName：要切换的目标目录。

实例

跳到 /usr/bin/ :

```
cd /usr/bin
```

跳到自己的 home 目录 :

```
cd ~
```

跳到目前目录的上上两层 :

```
cd ../..
```

Linux df命令

Linux df命令用于显示目前在Linux系统上的文件系统的磁盘使用情况统计。

语法

```
df [选项]... [FILE]...
```

- 文件-a, --all 包含所有的具有 0 Blocks 的文件系统
- 文件--block-size={SIZE} 使用 {SIZE} 大小的 Blocks
- 文件-h, --human-readable 使用人类可读的格式(预设值是不加这个选项的...)
- 文件-H, --si 很像 -h, 但是用 1000 为单位而不是用 1024
- 文件-i, --inodes 列出 inode 资讯，不列出已使用 block
- 文件-k, --kilobytes 就像是 --block-size=1024
- 文件-l, --local 限制列出的文件结构
- 文件-m, --megabytes 就像 --block-size=1048576
- 文件--no-sync 取得资讯前不 sync (预设值)
- 文件-P, --portability 使用 POSIX 输出格式
- 文件--sync 在取得资讯前 sync
- 文件-t, --type=TYPE 限制列出文件系统的 TYPE
- 文件-T, --print-type 显示文件系统的形式
- 文件-x, --exclude-type=TYPE 限制列出文件系统不要显示 TYPE
- 文件-v (忽略)
- 文件--help 显示这个帮手并且离开
- 文件--version 输出版本资讯并且离开

实例

显示文件系统的磁盘使用情况统计：

```
# df
Filesystem     1K-blocks    Used     Available Use% Mounted on
/dev/sda6       29640780 4320704     23814388  16%     /
udev             1536756       4     1536752    1%     /dev
tmpfs             617620     888     616732     1%     /run
none                5120       0     5120       0%     /run/lock
none             1544044     156     1543888    1%     /run/shm
```

第一列指定文件系统的名称，第二列指定一个特定的文件系统1K-块1K是1024字节为单位的总内存。用和可用列正在使用中，分别指定的内存量。

使用列指定使用的内存的百分比，而最后一栏"安装在"指定的文件系统的挂载点。

df也可以显示磁盘使用的文件系统信息：

```
# df test
Filesystem     1K-blocks    Used      Available Use% Mounted on
/dev/sda6       29640780    4320600   23814492  16%       /
```

用一个-i选项的df命令的输出显示inode信息而非块使用量。

```
df -i
Filesystem      Inodes    IUsed    IFree     IUse% Mounted on
/dev/sda6      1884160    261964   1622196   14%        /
udev           212748     560      212188    1%         /dev
tmpfs          216392     477      215915    1%         /run
none           216392     3        216389    1%         /run/lock
none           216392     8        216384    1%         /run/shm
```

显示所有的信息:

```
# df --total
Filesystem     1K-blocks    Used    Available Use% Mounted on
/dev/sda6       29640780 4320720    23814372  16%     /
udev             1536756       4    1536752   1%      /dev
tmpfs             617620     892    616728    1%      /run
none                5120       0    5120      0%      /run/lock
none             1544044     156    1543888   1%      /run/shm
total           33344320 4321772    27516860  14%
```

我们看到输出的末尾，包含一个额外的行，显示总的每一列。

-h选项，通过它可以产生可读的格式df命令的输出：

```
# df -h
Filesystem      Size  Used   Avail Use% Mounted on
/dev/sda6       29G   4.2G   23G   16%     /
udev            1.5G  4.0K   1.5G   1%     /dev
tmpfs           604M  892K   603M   1%     /run
none            5.0M     0   5.0M   0%     /run/lock
none            1.5G  156K   1.5G   1%     /run/shm
```

我们可以看到输出显示的数字形式的'G'（千兆字节），"M"（兆字节）和"K"（千字节）。

这使输出容易阅读和理解，从而使显示可读的。请注意，第二列的名称也发生了变化，为了使显示可读的"大小"。

Linux dirs命令

Linux dirs命令用于显示目录记录。

显示目录堆叠中的记录。

语法

```
dirs [+/-n -l]
```

参数：

- +n 显示从左边算起第n笔的目录。
- -n 显示从右边算起第n笔的目录。
- -l 显示目录完整的记录。

实例

列出"/home/cc/Ruijie"里所有内容的详细信息。可用如下命令。

```
dir -l /home/cc/Ruijie
```

下面是显示的内容：

```
$ dir -l /home/cc/Ruijie

总计2168

-rwxr-xr-x 1 cc cc  112876 2008-06-26 libpcap.so.0.6.2 -rwxr-xr-x 1 cc cc  737192 2008-06

-rwxr-xr-x 1 cc cc 1350772 2005-08-31 xrgsu
```

Linux du命令

Linux du命令用于显示目录或文件的大小。

du会显示指定的目录或文件所占用的磁盘空间。

语法

```
du [-abcDhHklmsSx][-L <符号连接>][-X <文件>][--block-size][--exclude=<目录或文件>][--max-dept
```

参数说明：

- -a或-all 显示目录中个别文件的大小。
- -b或-bytes 显示目录或文件大小时，以byte为单位。
- -c或--total 除了显示个别目录或文件的大小外，同时也显示所有目录或文件的总和。
- -D或--dereference-args 显示指定符号连接的源文件大小。
- -h或--human-readable 以K，M，G为单位，提高信息的可读性。
- -H或--si 与-h参数相同，但是K，M，G是以1000为换算单位。
- -k或--kilobytes 以1024 bytes为单位。
- -l或--count-links 重复计算硬件连接的文件。
- -L<符号连接>或--dereference<符号连接> 显示选项中所指定符号连接的源文件大小。
- -m或--megabytes 以1MB为单位。
- -s或--summarize 仅显示总计。
- -S或--separate-dirs 显示个别目录的大小时，并不含其子目录的大小。
- -x或--one-file-xystem 以一开始处理时的文件系统为准，若遇上其它不同的文件系统目录则略过。
- -X<文件>或--exclude-from=<文件> 在<文件>指定目录或文件。
- --exclude=<目录或文件> 略过指定的目录或文件。
- --max-depth=<目录层数> 超过指定层数的目录后，予以忽略。
- --help 显示帮助。
- --version 显示版本信息。

实例

显示目录或者文件所占空间:

```
# du
608     ./test6
308     ./test4
4       ./scf/lib
4       ./scf/service/deploy/product
4       ./scf/service/deploy/info
12      ./scf/service/deploy
16      ./scf/service
4       ./scf/doc
4       ./scf/bin
32      ./scf
8       ./test3
1288    .
```

只显示当前目录下面的子目录的目录大小和当前目录的总的大小，最下面的1288为当前目录的总大小

显示指定文件所占空间

```
# du log2012.log
300     log2012.log
```

方便阅读的格式显示test目录所占空间情况：

```
# du -h test
608K    test/test6
308K    test/test4
4.0K    test/scf/lib
4.0K    test/scf/service/deploy/product
4.0K    test/scf/service/deploy/info
12K     test/scf/service/deploy
16K     test/scf/service
4.0K    test/scf/doc
4.0K    test/scf/bin
32K     test/scf
8.0K    test/test3
1.3M    test
```

Linux edquota命令

Linux edquota命令用于编辑用户或群组的磁盘配额。

edquota预设会使用vi来编辑使用者或群组的磁盘配额设置。

语法

```
edquota [-p <源用户名称>][-ug][用户或群组名称...]
```

或

```
edquota [-ug] -t
```

参数：

- -u 设置用户的磁盘配额，这是预设的参数。
- -g 设置群组的磁盘配额。
- -p<源用户名称> 将源用户的磁盘配额设置套用至其他用户或群组。
- -t 设置宽限期限。

Linux mlabel命令

Linux mlabel命令用于设定磁盘的标签 (Label)。

如果磁盘上设定过标签，mlabel 会将他显示给使用者。如果没有指定新标签并且没有指定 c 或 s 选项，mlabel 会提示使用者输入新的标签。如果直接按下 Enter， 就会将原本的标签删除。

语法

```
mlabel [-vcs] drive:[new_label]
```

参数说明：

- -v 更多的讯息。
- -c 清除原有的标签，不出现提示讯息。
- -s 显示目前的标签，不出现提示讯息。

实例

将 A 盘的标签更改为 newlabel。

```
mlabel a:newlabel
```

Linux mkdir命令

Linux mkdir命令用于建立名称为 dirName 之子目录。

语法

```
mkdir [-p] dirName
```

参数说明：

- -p 确保目录名称存在，不存在的就建一个。

实例

在工作目录下，建立一个名为 AAA 的子目录：

```
mkdir AAA
```

在工作目录下的 BBB 目录中，建立一个名为 Test 的子目录。 若 BBB 目录原本不存在，则建立一个。（注：本例若不加 -p，且原本 BBB目录不存在，则产生错误。）

```
mkdir -p BBB/Test
```

Linux mdu命令

Linux mdu命令用于显示MS-DOS目录所占用的磁盘空间。

mdu为mstools工具指令，可显示MS-DOS文件系统中目录所占用的磁盘空间。

语法

```
mdu [-as][目录]
```

参数说明：

- -a 显示每个文件及整个目录所占用的空间。
- -s 仅显示整个目录所占用的空间。

Linux mdeltree命令

Linux mdeltree命令可用来删除 MSDOS 格式档案及目录。

mdeltree 会将所指定的目录与目录之下的所有档案与目录都删除掉。如果所指定的档案或目录不存在，则会传回错误讯息。

语法

```
mdeltree [-v] msdosdirectory [msdosdirectories...]
```

参数说明：

- -v 显示更多的信息。

实例

将 A 磁盘根目录中的 msdosdir 目录以下的档案与目录都删除掉。

```
mcopy a:msdosdir
```

Linux mcd命令

Linux mcd为mtools工具指令，可在MS-DOS文件系统中切换工作目录。若不加任何参数，则显示目前所在的磁盘与工作目录。

语法

```
mcd [msdosdirectory]
```

实例

变更目前工作目录到 a: emp 中。

```
mcd a: emp
```

传回目前工作目录。

```
mcd
```

Linux eject命令

Linux eject命令用于退出抽取式设备。

若设备已挂入，则eject会先将该设备卸除再退出。

语法

```
eject [-dfhnqrstv][-a <开关>][-c <光驱编号>][设备]
```

参数说明：

- [设备] 设备可以是驱动程序名称，也可以是挂入点。
- -a<开关>或--auto<开关> 控制设备的自动退出功能。
- -c<光驱编号>或--changerslut<光驱编号> 选择光驱柜中的光驱。
- -d或--default 显示预设的设备，而不是实际执行动作。
- -f或--floppy 退出抽取式磁盘。
- -h或--help 显示帮助。
- -n或--noop 显示指定的设备。
- -q或--tape 退出磁带。
- -r或--cdrom 退出光盘。
- -s或--scsi 以SCSI指令来退出设备。
- -t或--trayclose 关闭光盘的托盘。
- -v或--verbose 执行时，显示详细的说明。

实例

```
# eject //不加参数默认弹出
# eject -r /dev/cdrom //指定设备
```

Linux mount命令

Linux mount命令是经常会使用到的命令，它用于挂载Linux系统外的文件。

语法

```
mount [-hV]
mount -a [-fFnrsvw] [-t vfstype]
mount [-fnrsvw] [-o options [,...]] device | dir
mount [-fnrsvw] [-t vfstype] [-o options] device dir
```

参数说明：

- -V：显示程序版本
- -h：显示辅助讯息
- -v：显示较讯息，通常和 -f 用来除错。
- -a：将 /etc/fstab 中定义的所有档案系统挂上。
- -F：这个命令通常和 -a 一起使用，它会为每一个 mount 的动作产生一个行程负责执行。在系统需要挂上大量 NFS 档案系统时可以加快挂上的动作。
- -f：通常用在除错的用途。它会使 mount 并不执行实际挂上的动作，而是模拟整个挂上的过程。通常会和 -v 一起使用。
- -n：一般而言，mount 在挂上后会在 /etc/mtab 中写入一笔资料。但在系统中没有可写入档案系统存在的情况下可以用这个选项取消这个动作。
- -s-r：等于 -o ro
- -w：等于 -o rw
- -L：将含有特定标签的硬盘分割挂上。
- -U：将档案分割序号为 的档案系统挂下。-L 和 -U 必须在/proc/partition 这种档案存在时才有意义。
- -t：指定档案系统的型态，通常不必指定。mount 会自动选择正确的型态。
- -o async：打开非同步模式，所有的档案读写动作都会用非同步模式执行。
- -o sync：在同步模式下执行。
- -o atime、-o noatime：当 atime 打开时，系统会在每次读取档案时更新档案的『上一次调用时间』。当我们使用 flash 档案系统时可能会选项把这个选项关闭以减少写入的次数。
- -o auto、-o noauto：打开/关闭自动挂上模式。
- -o defaults:使用预设的选项 rw, suid, dev, exec, auto, nouser, and async.
- -o dev、-o nodev-o exec、-o noexec允许执行档被执行。
- -o suid、-o nosuid：
- 允许执行档在 root 权限下执行。
- -o user、-o nouser：使用者可以执行 mount/umount 的动作。

- -o remount：将一个已经挂下的档案系统重新用不同的方式挂上。例如原先是唯读的系统，现在用可读写的模式重新挂上。
- -o ro：用唯读模式挂上。
- -o rw：用可读写模式挂上。
- -o loop=：使用 loop 模式用来将一个档案当成硬盘分割挂上系统。

实例

将 /dev/hda1 挂在 /mnt 之下。

```
#mount /dev/hda1 /mnt
```

将 /dev/hda1 用唯读模式挂在 /mnt 之下。

```
#mount -o ro /dev/hda1 /mnt
```

将 /tmp/image.iso 这个光碟的 image 档使用 loop 模式挂在 /mnt/cdrom之下。用这种方法可以将一般网络上可以找到的 Linux 光 碟 ISO 档在不烧录成光碟的情况下检视其内容。

```
#mount -o loop /tmp/image.iso /mnt/cdrom
```

Linux mmd命令

Linux mmd命令用于在MS-DOS文件系统中建立目录。

mmd为mtools工具指令，模拟MS-DOS的md指令，可在MS-DOS的文件系统中建立目录。

语法

```
mmd [目录...]
```

Linux mrd命令

Linux mrd命令用于删除MS-DOS文件系统中的目录。

mrd为mtools工具指令，模拟MS-DOS的rd指令，可删除MS-DOS的目录。

语法

```
mrd [目录...]
```

Linux mzip命令

Linux mzip命令是Zip/Jaz磁盘驱动器控制指令。

mzip为mtools工具指令，可设置Zip或Jaz磁盘驱动区的保护模式以及执行退出磁盘的动作。

语法

```
mzip [-efpqruwx]
```

参数：

- -e 退出磁盘。
- -f 与-e参数一并使用，不管是否已经挂入磁盘中的文件系统，一律强制退出磁盘。
- -p 设置磁盘的写入密码。
- -q 显示目前的状态。
- -r 将磁盘设为防写状态。
- -u 退出磁盘以前，暂时解除磁盘的保护状态。
- -w 将磁盘设为可写入状态。
- -x 设置磁盘的密码。

Linux pwd命令

Linux pwd命令用于显示工作目录。

执行pwd指令可立刻得知您目前所在的工作目录的绝对路径名称。

语法

```
pwd [--help][--version]
```

参数说明:

- --help 在线帮助。
- --version 显示版本信息。

实例

查看当前所在目录：

```
# pwd
/root/test           #输出结果
```

Linux quota命令

Linux quota命令用于显示磁盘已使用的空间与限制。

执行quota指令，可查询磁盘空间的限制，并得知已使用多少空间。

语法

```
quota [-quvV][用户名称...] 或 quota [-gqvV][群组名称...]
```

参数说明：

- -g 列出群组的磁盘空间限制。
- -q 简明列表，只列出超过限制的部分。
- -u 列出用户的磁盘空间限制。
- -v 显示该用户或群组，在所有挂入系统的存储设备的空间限制。
- -V 显示版本信息。

实例

```
# quota -guvs    <==显示目前执行者（就是 root ）的 quota 值
# quota -uvs test <==显示 test 这个使用者的 quota 值
```

Linux mmount命令

Linux mmount命令用于挂入MS-DOS文件系统。

mmount为mtools工具指令，可根据[mount参数]中的设置，将磁盘内容挂入到Linux目录中。

语法

```
mmount [驱动器代号][mount参数]
```

参数：

- [mount参数]的用法请参考 mount指令。

Linux rmdir命令

Linux rmdir命令删除空的目录。

语法

```
rmdir [-p] dirName
```

参数：

- -p 是当子目录被删除后使它也成为空目录的话，则顺便一并删除。

实例

将工作目录下，名为 AAA 的子目录删除：

```
rmdir AAA
```

在工作目录下的 BBB 目录中，删除名为 Test 的子目录。若 Test 删除后，BBB 目录成为空目录，则 BBB 亦予删除。

```
rmdir -p BBB/Test
```

Linux rmt命令

Linux rmt命令通过进程间通信远程控制磁带机。

通过rmt指令，用户可通过IPC连线，远端操控磁带机的倾倒和还原操作。

语法

```
rmt
```

Linux stat命令

Linux stat命令用于显示inode内容。

stat以文字的格式来显示inode的内容。

语法

```
stat [文件或目录]
```

实例

查看 testfile 文件的inode内容内容，可以用以下命令：

stat testfile

执行以上命令输出结果：

```
# stat testfile                #输入命令
  File: `testfile'
  Size: 102             Blocks: 8          IO Block: 4096   regular file
Device: 807h/2055d      Inode: 1265161     Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2014-08-13 14:07:20.000000000 +0800
Modify: 2014-08-13 14:07:07.000000000 +0800
Change: 2014-08-13 14:07:07.000000000 +0800
```

Linux tree命令

Linux tree命令用于以树状图列出目录的内容。

执行tree指令，它会列出指定目录下的所有文件，包括子目录里的文件。

语法

```
tree [-aACdDfFgilnNpqstux][-I <范本样式>][-P <范本样式>][目录...]
```

参数说明：

- -a 显示所有文件和目录。
- -A 使用ASNI绘图字符显示树状图而非以ASCII字符组合。
- -C 在文件和目录清单加上色彩，便于区分各种类型。
- -d 显示目录名称而非内容。
- -D 列出文件或目录的更改时间。
- -f 在每个文件或目录之前，显示完整的相对路径名称。
- -F 在执行文件，目录，Socket，符号连接，管道名称名称，各自加上"*","/","=","@","|"号。
- -g 列出文件或目录的所属群组名称，没有对应的名称时，则显示群组识别码。
- -i 不以阶梯状列出文件或目录名称。
- -I<范本样式> 不显示符合范本样式的文件或目录名称。
- -l 如遇到性质为符号连接的目录，直接列出该连接所指向的原始目录。
- -n 不在文件和目录清单加上色彩。
- -N 直接列出文件和目录名称，包括控制字符。
- -p 列出权限标示。
- -P<范本样式> 只显示符合范本样式的文件或目录名称。
- -q 用"?"号取代控制字符，列出文件和目录名称。
- -s 列出文件或目录大小。
- -t 用文件和目录的更改时间排序。
- -u 列出文件或目录的拥有者名称，没有对应的名称时，则显示用户识别码。
- -x 将范围局限在现行的文件系统中，若指定目录下的某些子目录，其存放于另一个文件系统上，则将该子目录予以排除在寻找范围外。

实例

以树状图列出当前目录结构。可直接使用如下命令：

```
tree
```

该命令有如下输出结果：

```
# tree                   #以树状图列出当前目录结构
.                        #当前目录结构
|-- README
|-- examples.desktop
|-- file
|-- file.new
|-- index.htm
|-- test
| |-- README
| |-- file
| |-- testfile
| |-- testfile1
| |-- xaa
| |-- xab
| |-- xac
| |-- xad
| |-- xae
| |-- xaf
| |-- xag
| |-- xah
| `-- xai
|-- test.tar.gz
|-- test.zip
|-- testfile
|-- testfile.new
|-- testfile.patch
|-- testfile1
|-- testfile2
|-- testfile3
|-- xaa
|-- xab
|-- xac
|-- xad
|-- xae
|-- xaf
|-- xag
|-- xah
|-- xai
|-- \345\205\254\345\205\261\347\232\204
|-- \345\233\276\347\211\207
| |-- 075b5c2bb1628c1a5343c10a.jpg
| |-- 0c978fe989ac787e799757095719d3c4.jpg
| |-- 20050726194826866443.jpg
| |-- 20061113171548785122.jpg
| |-- 2007102221576687.jpg
| |-- 39.jpg
| |-- 434887ec4340916a78f0559a.jpg
| |-- 498da016ac02fb2bc93d6d08.jpg
| |-- 7b284f5a0f854da2f3bf90b204149a34.jpg
| |-- 9196c030d342a68d5edf0e98.jpg
| |-- a56c5a90de15c8a9a977a4cc.jpg
| |-- c74f62167c9d2b244a90a79e.jpg
| `-- img13.jpg
|-- \346\226\207\346\241\243
|-- \346\241\214\351\235\242
|-- \346\250\241\346\235\277
|-- \350\247\206\351\242\221
`-- \351\237\263\344\271\220
8 directories, 48 files           #统计信息，该目录共8个子目录，48个文件
```

Linux umount命令

Linux umount命令用于卸除文件系统。

umount可卸除目前挂在Linux目录中的文件系统。

语法

```
umount [-ahnrvV][-t <文件系统类型>][文件系统]
```

参数：

- -a 卸除/etc/mtab中记录的所有文件系统。
- -h 显示帮助。
- -n 卸除时不要将信息存入/etc/mtab文件中。
- -r 若无法成功卸除，则尝试以只读的方式重新挂入文件系统。
- -t<文件系统类型> 仅卸除选项中所指定的文件系统。
- -v 执行时显示详细的信息。
- -V 显示版本信息。
- [文件系统] 除了直接指定文件系统外，也可以用设备名称或挂入点来表示文件系统。

实例

下面两条命令分别通过设备名和挂载点卸载文件系统，同时输出详细信息：

```
# umount -v /dev/sda1          通过设备名卸载
/dev/sda1 umounted
# umount -v /mnt/mymount/      通过挂载点卸载
/tmp/diskboot.img umounted
```

如果设备正忙，卸载即告失败。卸载失败的常见原因是，某个打开的shell当前目录为挂载点里的某个目录：

```
# umount -v /mnt/mymount/
umount: /mnt/mymount: device is busy
umount: /mnt/mymount: device is busy
```

Linux ls命令

Linux ls命令用于显示指定工作目录下之内容（列出目前工作目录所含之文件及子目录)。

语法

```
ls [-alrtAFR] [name...]
```

参数：

- -a 显示所有文件及目录 (ls内定将文件名或目录名称开头为"."的视为隐藏档，不会列出)
- -l 除文件名称外，亦将文件型态、权限、拥有者、文件大小等资讯详细列出
- -r 将文件以相反次序显示(原定依英文字母次序)
- -t 将文件依建立时间之先后次序列出
- -A 同 -a，但不列出 "." (目前目录) 及 ".." (父目录)
- -F 在列出的文件名称后加一符号；例如可执行档则加 "*", 目录则加 "/"
- -R 若目录下有文件，则以下之文件亦皆依序列出

实例

列出根目录()下的所有目录：

```
# ls /
bin               dev   lib         media  net   root     srv  upload  www
boot              etc   lib64       misc   opt   sbin     sys  usr
home  lost+found  mnt    proc  selinux  tmp  var
```

列出目前工作目录下所有名称是 s 开头的文件，越新的排越后面：

```
ls -ltr s*
```

将 /bin 目录以下所有目录及文件详细资料列出：

```
ls -lR /bin
```

列出目前工作目录下所有文件及目录；目录于名称后加 "/", 可执行档于名称后加 "*"：

```
ls -AF
```

Linux quotacheck命令

Linux quotacheck命令用于检查磁盘的使用空间与限制。

执行quotacheck指令，扫描挂入系统的分区，并在各分区的文件系统根目录下产生quota.user和quota.group文件，设置用户和群组的磁盘空间限制。

语法

```
quotacheck [-adgRuv][文件系统...]
```

参数：

- -a 扫描在/etc/fstab文件里，有加入quota设置的分区。
- -d 详细显示指令执行过程，便于排错或了解程序执行的情形。
- -g 扫描磁盘空间时，计算每个群组识别码所占用的目录和文件数目。
- -R 排除根目录所在的分区。
- -u 扫描磁盘空间时，计算每个用户识别码所占用的目录和文件数目。
- -v 显示指令执行过程。

Linux quotaoff命令

Linux quotaoff命令关闭磁盘空间限制。

执行quotaoff指令可关闭用户和群组的磁盘空间限制。

语法

```
quotaoff [-aguv][文件系统...]
```

参数说明：

- -a 关闭在/etc/fstab文件里，有加入quota设置的分区的空间限制。
- -g 关闭群组的磁盘空间限制。
- -u 关闭用户的磁盘空间限制。
- -v 显示指令执行过程。

实例

关闭配额限制:

```
# quotaoff -a
```

Linux lndir命令

Linux lndir命令用于连接目录内容。

执行lndir指令，可一口气把源目录底下的文件和子目录统统建立起相互对应的符号连接。

语法

```
lndir [-ignorelinks][-silent][源目录][目的目录]
```

参数：

- -ignorelinks 直接建立符号连接的符号连接。
- -silent 不显示指令执行过程。

实例

给目录下所有的文件或者子文件目录建立链接：

```
lndir /home/uptech abc
```

Linux repquota命令

Linux repquota命令用于检查磁盘空间限制的状态。

执行repquota指令，可报告磁盘空间限制的状况，清楚得知每位用户或每个群组已使用多少空间。

语法

```
repquota [-aguv][文件系统...]
```

参数说明：

- -a 列出在/etc/fstab文件里，有加入quota设置的分区的使用状况，包括用户和群组。
- -g 列出所有群组的磁盘空间限制。
- -u 列出所有用户的磁盘空间限制。
- -v 显示该用户或群组的所有空间限制。

Linux quotaon命令

Linux quotaon命令用于开启磁盘空间限制。

执行quotaon指令可开启用户和群组的才磅秒年空间限制，各分区的文件系统根目录必须有quota.user和quota.group配置文件。

语法

```
quotaon [-aguv][文件系统...]
```

参数说明：

- -a 开启在/ect/fstab文件里，有加入quota设置的分区的空间限制。
- -g 开启群组的磁盘空间限制。
- -u 开启用户的磁盘空间限制。
- -v 显示指令指令执行过程。

Linux命令大全 - 磁盘维护

badblocks	cfdisk	dd	e2fsck
ext2ed	fsck	fsck.minix	fsconf
fdformat	hdparm	mformat	mkbootdisk
mkdosfs	mke2fs	mkfs.ext2	mkfs.msdos
mkinitrd	mkisofs	mkswap	mpartition
swapon	symlinks	sync	mbadblocks
mkfs.minix	fsck.ext2	fdisk	losetup
mkfs	sfdisk	swapoff	

Linux badblocks命令

Linux badblocks命令用于检查磁盘装置中损坏的区块。

执行指令时须指定所要检查的磁盘装置，及此装置的磁盘区块数。

语法

```
badblocks [-svw][-b <区块大小>][-o <输出文件>][磁盘装置][磁盘区块数][启始区块]
```

参数说明：

- -b<区块大小> 指定磁盘的区块大小，单位为字节。
- -o<输出文件> 将检查的结果写入指定的输出文件。
- -s 在检查时显示进度。
- -v 执行时显示详细的信息。
- -w 在检查时，执行写入测试。
- [磁盘装置] 指定要检查的磁盘装置。
- [磁盘区块数] 指定磁盘装置的区块总数。
- [启始区块] 指定要从哪个区块开始检查。

实例

查看系统当前硬盘信息。

```
# fdisk -l
```

例如，显示信息如下：

```
Disk /dev/sda: 298.9 GB, 298999349248 bytes
255 heads, 63 sectors/track, 36351 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         262     2104483+  82  Linux swap / Solaris
/dev/sda2   *         263       32898   262148670   83  Linux
/dev/sda3           32899       36351    27736222+  83  Linux

Disk /dev/sdb: 42.9 GB, 42949672960 bytes
64 heads, 32 sectors/track, 40960 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
```

通过命令扫描硬盘。

```
# badblocks -s -v /dev/sdnx
```

其中n表示硬盘设备名，x表示硬盘对应的分区号。例如需要检查"/dev/sda2"，执行命令如下：

```
# badblocks -s -v /dev/sda2

Checking blocks 0 to 30681000
Checking for bad blocks (read-only test): 306809600674112/ 306810000000
30680964
30680965
30680966
30680967
30680968
30680969
30680970
30680971
30680972
30680973
...
done
Pass completed, 37 bad blocks found.其中，“37 bad blocks found”表示硬盘存在37个坏块。
```

Linux cfdisk命令

Linux cfdisk命令用于磁盘分区。

cfdisk是用来磁盘分区的程序，它十分类似DOS的fdisk，具有互动式操作界面而非传统fdisk的问答式界面，您可以轻易地利用方向键来操控分区操作。

语法

```
cfdisk [-avz][-c <柱面数目>-h <磁头数目>-s <盘区数目>][-P <r,s,t>][外围设备代号]
```

参数说明：

- -a 在程序里不用反白代表选取，而以箭头表示。
- -c<柱面数目> 忽略BIOS的数值，直接指定磁盘的柱面数目。
- -h<磁头数目> 忽略BIOS的数值，直接指定磁盘的磁头数目。
- -P<r,s,t> 显示分区表的内容，附加参数"r"会显示整个分区表的详细资料，附加参数"s"会依照磁区的顺序显示相关信息，附加参数"t"则会以磁头，磁区，柱面的方式来显示资料。
- -s<磁区数目> 忽略BIOS的数值，直接指定磁盘的磁区数目。
- -v 显示版本信息。
- -z 不读取现有的分区，直接当作没有分区的新磁盘使用。

实例

进行磁盘分区：

```
# cfsik
```

进行磁盘分区，使用箭头进行操作，而不使用反白表示：

```
# cfsik -a
```

进行磁盘分区，使用箭头进行操作，而不使用反白表示：

```
# cfsik -s 3
```

VM59220:1 Resource interpreted as Image but transferred with MIME type text/html: "http://googleads.g.doubleclick.net/pagead/adview?ai=C9xWpIRctVK75CYm28gXc8Y…bC1FEwdf3NGxgwxaMy7yS-2Ac7otU34AGyoyh--HWiPSLAaAGIQ&sigh=-A-s3VVItog&vis=1". ads?client=ca-pub-5751451760833794

Linux dd命令

Linux dd命令用于读取、转换并输出数据。

dd可从标准输入或文件中读取数据，根据指定的格式来转换数据，再输出到文件、设备或标准输出。

参数说明:

- if=文件名：输入文件名，缺省为标准输入。即指定源文件。
- of=文件名：输出文件名，缺省为标准输出。即指定目的文件。
- ibs=bytes：一次读入bytes个字节，即指定一个块大小为bytes个字节。
  obs=bytes：一次输出bytes个字节，即指定一个块大小为bytes个字节。
  bs=bytes：同时设置读入/输出的块大小为bytes个字节。
- cbs=bytes：一次转换bytes个字节，即指定转换缓冲区大小。
- skip=blocks：从输入文件开头跳过blocks个块后再开始复制。
- seek=blocks：从输出文件开头跳过blocks个块后再开始复制。
- count=blocks：仅拷贝blocks个块，块大小等于ibs指定的字节数。
- conv=<关键字>，关键字可以有以下11种：
    - conversion：用指定的参数转换文件。
    - ascii：转换ebcdic为ascii
    - ebcdic：转换ascii为ebcdic
    - ibm：转换ascii为alternate ebcdic
    - block：把每一行转换为长度为cbs，不足部分用空格填充
    - unblock：使每一行的长度都为cbs，不足部分用空格填充
    - lcase：把大写字符转换为小写字符
    - ucase：把小写字符转换为大写字符
    - swab：交换输入的每对字节
    - noerror：出错时不停止
    - notrunc：不截短输出文件
    - sync：将每个输入块填充到ibs个字节，不足部分用空（NUL）字符补齐。
- --help：显示帮助信息
- --version：显示版本信息

实例

在Linux 下制作启动盘，可使用如下命令：

```
dd if=boot.img of=/dev/fd0 bs=1440k
```

将testfile文件中的所有英文字母转换为大写，然后转成为testfile_1文件，在命令提示符中使用如下命令：

```
dd if=testfile_2 of=testfile_1 conv=ucase
```

其中testfile_2 的内容为：

```
$ cat testfile_2 #testfile_2的内容
HELLO LINUX!
Linux is a free unix-type opterating system.
This is a linux testfile!
Linux test
```

转换完成后，testfile_1 的内容如下：

```
$ dd if=testfile_2 of=testfile_1 conv=ucase #使用dd 命令，大小写转换记录了0+1 的读入
记录了0+1 的写出
95字节（95 B）已复制，0.000131446 秒，723 KB/s
cmd@hdd-desktop:~$ cat testfile_1 #查看转换后的testfile_1文件内容
HELLO LINUX!
LINUX IS A FREE UNIX-TYPE OPTERATING SYSTEM.
THIS IS A LINUX TESTFILE!
LINUX TEST #testfile_2中的所有字符都变成了大写字母
```

由标准输入设备读入字符串，并将字符串转换成大写后，再输出到标准输出设备，使用的命令为：

```
dd conv=ucase
```

输入以上命令后按回车键，输入字符串，再按回车键，按组合键Ctrl+D 退出，出现以下结果：

```
$ dd conv=ucase
Hello Linux! #输入字符串后按回车键
HELLO LINUX! #按组合键Ctrl+D退出，转换成大写结果
记录了0+1 的读入
记录了0+1 的写出
13字节（13 B）已复制，12.1558 秒，0.0 KB/s
```

href http://www.w3cschool.cc/linux/linux-comm-e2fsck.html linux-comm-e2fsck.html:31 'Attr.nodeValue' is deprecated. Please use 'value' instead. adsbygoogle.js:32

Linux e2fsck命令

Linux e2fsck命令用于检查使用 Linux ext2 档案系统的 partition 是否正常工作。

语法

```
e2fsck [-pacnydfvFV] [-b superblock] [-B blocksize] [-l|-L bad_blocks_file] [-C fd] devic
```

参数说明：

- device ： 预备检查的硬盘 partition，例如：/dev/sda1
- -a : 对 partition 做检查，若有问题便自动修复，等同 -p 的功能
- -b : 设定存放 superblock 的位置
- -B : 设定单位 block 的大小
- -c : 检查该partition 是否有坏轨
- -C file : 将检查的结果存到 file 中以便查看
- -d : 列印 e2fsck 的 debug 结果
- -f : 强制检查
- -F : 在开始检查前，将device 的 buffer cache 清空，避免有错误发生
- -l bad_blocks_file : 将有坏轨的block资料加到 bad_blocks_file 里面
- -L bad_blocks_file : 设定坏轨的block资料存到 bad_blocks_file 里面，若无该档则自动产生
- -n : 将档案系统以[唯读]方式开启
- -p : 对 partition 做检查，若有问题便自动修复
- -v : 详细显示模式
- -V : 显示出目前 e2fsck 的版本
- -y : 预先设定所有检查时的问题均回答[是]

实例

检查 /dev/hda5 是否正常，如果有异常便自动修复，并且设定若有问答，均回答[是] :

```
e2fsck -a -y /dev/hda5
```

注意 :

大部份使用 e2fsck 来检查硬盘 partition 的情况时，通常都是情形特殊，因此最好先将该 partition umount，然后再执行 e2fsck 来做检查，若是要非要检查 / 时，则请进入 singal user mode 再执行。

Linux ext2ed命令

Linux ext2ed命令是ext2文件系统编辑程序。

ext2ed可直接处理硬盘分区上的数据，这指令只有Red Hat Linux才提供。

语法

```
ext2ed
```

一般指令：

- setdevice[设备名称] 指定要处理的设备。
- disablewrite 将ext2ed设为只读的状态。
- enablewrite 将ext2ed设为可读写的状态。
- help[指令] 显示个别指令的帮助。
- next 移至下一个单位，单位会依目前所在的模式而异。
- prev 移至前一个单位，单位会依目前所在的模式而异。
- pgup 移至下一页。
- pgdn 移至上一页。
- set 修改目前的数据，参数会依目前所在的模式而异。
- writedata 在执行此指令之后，才会实际修改分区中的数据。
- ext2进入3种模式的指令
- super 进入main superblock,即Superblock模式。
- group<编号> 进入指定的group，即Group模式。
- cd<目录或文件> 在inode模式下，进入指定的目录或文件，即Inode模式。
- Superblock模式
- gocopy<备份编号> 进入指定的superblock备份。
- setactivecopy 将目前所在的superblock，复制到main superblock。
- Group模式
- blockbitmap 显示目前groupo的区块图。
- inode 进入目前group的第一个inode。
- inodebitmap 显示目前group的inode二进制码。
- Inode模式
- dir 进入目录模式。
- file 进入文件模式。

Linux mkbootdisk命令

Linux mkbootdisk命令用于建立目前系统的启动盘。

mkbootdisk可建立目前系统的启动盘。

语法

```
mkbootdisk [--noprompt][--verbose][--version][--device <设备>][--mkinitrdargs <参数>][kern
```

参数：

- --device<设备> 指定设备。
- --mkinitrdargs<参数> 设置mkinitrd的参数。
- --noprompt 不会提示用户插入磁盘。
- --verbose 执行时显示详细的信息。
- --version 显示版本信息。

Linux fsck命令

Linux fsck命令用于检查与修复 Linux 档案系统，可以同时检查一个或多个 Linux 档案系统。

语法

```
fsck [-sACVRP] [-t fstype] [--] [fsck-options] filesys [...]
```

参数 ：

- filesys ： device 名称(eg./dev/sda1)，mount 点 (eg. / 或 /usr)
- -t : 给定档案系统的型式，若在 /etc/fstab 中已有定义或 kernel 本身已支援的则不需加上此参数
- -s : 依序一个一个地执行 fsck 的指令来检查
- -A : 对/etc/fstab 中所有列出来的 partition 做检查
- -C : 显示完整的检查进度
- -d : 列印 e2fsck 的 debug 结果
- -p : 同时有 -A 条件时，同时有多个 fsck 的检查一起执行
- -R : 同时有 -A 条件时，省略 / 不检查
- -V : 详细显示模式
- -a : 如果检查有错则自动修复
- -r : 如果检查有错则由使用者回答是否修复

实例

检查 msdos 档案系统的 /dev/hda5 是否正常，如果有异常便自动修复 :

```
fsck -t msdos -a /dev/hda5
```

注意 此指令可与 /etc/fstab 相互参考操作来加以了解。

Linux fsck.minix命令

Linux fsck.minix命令用于检查文件系统并尝试修复错误。

当minix文件系统发生错误时，可用fsck.minix指令尝试加以参考。

语法

```
fsck.minix [-aflmrsv][外围设备代号]
```

参数：

- -a 自动修复文件系统，不询问任何问题。
- -f 强制对该文件系统进行完整检查，纵然该文件系统在概略检查下没有问题。
- -l 列出所有文件名称。
- -m 使用类似MINIX操作系统的警告信息。
- -r 采用互动模式，在执行修复时询问问题，让用户得以确认并决定处理方式。
- -s 显示该分区第一个磁区的相关信息。
- -v 显示指令执行过程。

Linux fsconf命令

Linux fsconf命令用于设置文件系统相关功能。

fsconf是Red Hat Linux发行版专门用来调整Linux各项设置的程序。

语法

```
fsconf [--check]
```

参数：

- --chedk 检查特定文件的权限。

Linux fdformat命令

Linux fdformat命令用于对指定的软碟机装置进行低阶格式化。

使用这个指令对软碟格式化的时候，最好指定像是下面的装置：

- /dev/fd0d360 磁碟机 A:，磁片为 360KB 磁碟
- /dev/fd0h1440 磁碟机 A:，磁片为 1.4MB 磁碟
- /dev/fd1h1200 磁碟机 B:，磁片为 1.2MB 磁碟

如果使用像是 /dev/fd0 之类的装置，如果里面的磁碟不是标准容量，格式化可能会失败。在这种情况之下，使用者可以用 setfdprm 指令先行指定必要参数。

语法

```
fdformat [-n] device
```

参数：

- -n 关闭确认功能。这个选项会关闭格式化之后的确认步骤。

实例

```
fdformat -n /dev/fd0h1440
```

将磁碟机 A 的磁片格式化成 1.4MB 的磁片。并且省略确认的步骤。

Linux hdparm命令

Linux hdparm命令用于显示与设定硬盘的参数。

hdparm可检测，显示与设定IDE或SCSI硬盘的参数。

语法

```
hdparm [-CfghiIqtTvyYZ][-a <快取分区>][-A <0或1>][-c <I/O模式>][-d <0或1>][-k <0或1>][-K <0
```

参数说明：

- -a<快取分区> 设定读取文件时，预先存入块区的分区数，若不加上<快取分区>选项，则显示目前的设定。
- -A<0或1> 启动或关闭读取文件时的快取功能。
- -c<I/O模式> 设定IDE32位I/O模式。
- -C 检测IDE硬盘的电源管理模式。
- -d<0或1> 设定磁盘的DMA模式。
- -f 将内存缓冲区的数据写入硬盘，并清楚缓冲区。
- -g 显示硬盘的磁轨，磁头，磁区等参数。
- -h 显示帮助。
- -i 显示硬盘的硬件规格信息，这些信息是在开机时由硬盘本身所提供。
- -I 直接读取硬盘所提供的硬件规格信息。
- -k<0或1> 重设硬盘时，保留-dmu参数的设定。
- -K<0或1> 重设硬盘时，保留-APSWXZ参数的设定。
- -m<磁区数> 设定硬盘多重分区存取的分区数。
- -n<0或1> 忽略硬盘写入时所发生的错误。
- -p<PIO模式> 设定硬盘的PIO模式。
- -P<磁区数> 设定硬盘内部快取的分区数。
- -q 在执行后续的参数时，不在屏幕上显示任何信息。
- -r<0或1> 设定硬盘的读写模式。
- -S<时间> 设定硬盘进入省电模式前的等待时间。
- -t 评估硬盘的读取效率。
- -T 平谷硬盘快取的读取效率。
- -u<0或1> 在硬盘存取时，允许其他中断要求同时执行。
- -v 显示硬盘的相关设定。
- -W<0或1> 设定硬盘的写入快取。
- -X<传输模式> 设定硬盘的传输模式。

- -y 使IDE硬盘进入省电模式。
- -Y 使IDE硬盘进入睡眠模式。
- -Z 关闭某些Seagate硬盘的自动省电功能。

实例

显示硬盘的相关设置：

```
# hdparm /dev/sda
 /dev/sda:
 IO_support = 0 (default 16-bit)
 readonly = 0 (off)
 readahead = 256 (on)
 geometry = 19929［柱面数］/255［磁头数］/63［扇区数］, sectors = 320173056［总扇区数］, start =
```

显示硬盘的柱面、磁头、扇区数

```
# hdparm -g /dev/sda
 /dev/sda:
 geometry = 19929［柱面数］/255［磁头数］/63［扇区数］, sectors = 320173056［总扇区数］, start =
```

评估硬盘的读取效率

```
 hdparm -t /dev/sda
 /dev/sda:
 Timing buffered disk reads: 166 MB in 3.03 seconds = 54.85 MB/sec
 [root@linuxso.com ~]# hdparm -t /dev/sda
 /dev/sda:
 Timing buffered disk reads: 160 MB in 3.01 seconds = 53.11 MB/sec
 [root@linuxso.com ~]# hdparm -t /dev/sda
 /dev/sda:
 Timing buffered disk reads: 166 MB in 3.00 seconds = 55.31 MB/sec
```

Linux mformat命令

Linux mformat命令用于对MS-DOS文件系统的磁盘进行格式化。

在已经做过低阶格式化的磁片上建立 DOS 档案系统。如果在编程 mtools 的时候把 USE_2M 的参数打开，部分与 2M 格式相关的参数就会发生作用。否则这些参数（像是 S,2,1,M）不会发生作用。

语法

```
mformat [-t cylinders] [-h heads] [-s sectors] [-l volume_label] [-F] [-I fsVer-sion] [-S
```

参数：

- -t 磁柱（synlider）数
- -h 磁头（head）数
- -s 每一磁轨的磁区数
- -l 标签
- -F 将磁碟格式化为 FAT32 格式，不过这个参数还在实验中。
- -I 设定 FAT32 中的版本号。这当然也还在实验中。
- -S 磁区大小代码，计算方式为 sector = 2^(大小代码+7)
- -c 磁丛（cluster）的磁区数。如果所给定的数字会导致磁丛数超过 FAT 表的限制，mformat 会自动放大磁区数。
- -s
- -M 软件磁区大小。这个数字就是系统回报的磁区大小。通常是和实际的大小相同。
- -a 如果加上这个参数，mformat 会产生一组 Atari 系统的序号给这块软碟。
- -X 将软碟格式化成 XDF 格式。使用前必须先用 xdfcopy 指令对软碟作低阶格式化的动作。
- -C 产生一个可以安装 MS-DOS 档案系统的磁碟影像档（disk image）。当然对一个实体磁碟机下这个参数是没有意义的。
- -H 隐藏磁区的数目。这通常适用在格式化硬盘的分割区时，因为通常一个分割区的前面还有分割表。这个参数未经测试，能不用就不用。
- -n 磁碟序号
- -r 根目录的大小，单位是磁区数。这个参数只对 FAT12 和 FAT16 有效。
- -B 使用所指定的档案或是设备的开机磁区做为这片磁片或分割区的开机磁区。当然当中的硬件参数会随之更动。
- -k 尽量保持原有的开机磁区。
- -0 第 0 轨的资料传输率

- -A 第 0 轨以外的资料传输率
- -2 使用 2m 格式
- -1 不使用 2m 格式

实例

用预设值把 a: （就是 /dev/fd0）里的磁碟片格式化。

```
mformat a:
```

Linux mkdosfs命令

Linux mkdosfs命令用于建立DOS文件系统。

device 指你想要建立 DOS 档案系统的装置代号。像是 /dev/hda1 等等。 block_count 则是你希望配置的区块数。如果 block_count 没有指定则系统会自动替你计算符合该装置大小的区块数。

```
mkdosfs [ -c | -l filename ]
    [ -f number_of_FATs ]
    [ -F FAT_size ]
    [ -i volume_id ]
    [ -m message_file ]
    [ -n volume_name ]
    [ -r root_dir_entry ]
    [ -s sector_per_cluster ]
    [ -v ]
    device
    [ block_count ]
```

参数：

- -c 建立档案系统之前先检查是否有坏轨。
- -l 从得定的档案中读取坏轨记录。
- -f 指定档案配置表（FAT , File Allocation Table)的数量。预设值为 2 。目前 Linux 的 FAT 档案系统不支援超过 2 个 FAT 表。通常这个不需要改。
- -F 指定 FAT 表的大小，通常是 12 或是 16 个位元组。12 位元组通常用于磁碟片，16 位元组用于一般硬盘的分割区，也就是所谓的 FAT16 格式。这个值通常系统会自己选定适当的值。在磁碟片上用 FAT16 通常不会发生作用，反之在硬盘上用 FAT12 亦然。
- -i 指定 Volume ID。一般是一个 4 个位元组的数字，像是 2e203a47 。如果不给系统会自己产生。
- -m 当使用者试图用这片磁片或是分割区开机，而上面没有操作系统时，系统会给使用者一段警告讯息。这个参数就是用来变更这个讯息的。你可以先用档案编辑好，然后用这个参数指定，或是用
- -m -
- 这样系统会要求你直接输入这段文字。要特别注意的是，档案里的字串长度不要超过 418 个字，包括展开的跳栏符号（TAB）和换行符号（换行符号在 DOS 底下算两个字元！）
- -n 指定 Volume Name，就是磁碟标签。如同在 DOS 底下的 format 指令一样，给不给都可以。没有预设值。
- -r 指定根目录底下的最大档案数。这里所谓的档案数包括目录。预设值是在软碟上是 112 或是 224 ，在硬盘上是 512。没事不要改这个数字。
- -s 每一个磁丛（cluster）的磁区数。必须是 2 的次方数。不过除非你知道你在作什么，这个值不要乱给。

- -v 提供额外的讯息

实例

将 A 槽里的磁碟片格式化为 DOS 格式，并将标签设为 Tester

```
mkdosfs -n Tester /dev/fd0
```

Linux mke2fs命令

Linux mke2fs命令用于建立ext2文件系统。

语法

```
mke2fs [-cFMqrSvV][-b <区块大小>][-f <不连续区段大小>][-i <字节>][-N <inode数>][-l <文件>][-L
```

参数：

- -b<区块大小> 指定区块大小，单位为字节。
- -c 检查是否有损坏的区块。
- -f<不连续区段大小> 指定不连续区段的大小，单位为字节。
- -F 不管指定的设备为何，强制执行mke2fs。
- -i<字节> 指定"字节/inode"的比例。
- -N<inode数> 指定要建立的inode数目。
- -l<文件> 从指定的文件中，读取文件西中损坏区块的信息。
- -L<标签> 设置文件系统的标签名称。
- -m<百分比值> 指定给管理员保留区块的比例，预设为5%。
- -M 记录最后一次挂入的目录。
- -q 执行时不显示任何信息。
- -r 指定要建立的ext2文件系统版本。
- -R=<区块数> 设置磁盘阵列参数。
- -S 仅写入superblock与group descriptors，而不更改inode able inode bitmap以及block bitmap。
- -v 执行时显示详细信息。
- -V 显示版本信息。

Linux mkfs.ext2命令

功能说明：与 mke2fs命令 相同

Linux mkfs.msdos命令

功能说明：与 mkdosfs 命令 相同。

Linux mkinitrd命令

Linux mkinitrd命令用于建立要载入ramdisk的映像文件。

mkinitrd可建立映像文件，以供Linux开机时载入ramdisk。

语法

```
mkinitrd [-fv][--omit-scsi-modules][--version][--preload=<模块名称>][--with=<模块名称>][映像
```

参数：

- -f 若指定的映像问家名称与现有文件重复，则覆盖现有的文件。
- -v执行时显示详细的信息。
- --omit-scsi-modules 不要载入SCSI模块。
- --preload=<模块名称> 指定要载入的模块。
- --with=<模块名称> 指定要载入的模块。
- --version 显示版本信息。

Linux mkisofs命令

Linux mkisofs命令用于建立ISO 9660映像文件。

mkisofs可将指定的目录与文件做成ISO 9660格式的映像文件，以供刻录光盘。

语法

```
mkisofs [-adDfhJlLNrRTvz][-print-size][-quiet][-A <应用程序ID>][-abstract <摘要文件>][-b <开
```

参数：

- -a或--all mkisofs通常不处理备份文件。使用此参数可以把备份文件加到映像文件中。
- -A<应用程序ID>或-appid<应用程序ID> 指定光盘的应用程序ID。
- -abstract<摘要文件> 指定摘要文件的文件名。
- -b<开机映像文件>或-eltorito-boot<开机映像文件> 指定在制作可开机光盘时所需的开机映像文件。
- -biblio<ISBN文件> 指定ISBN文件的文件名，ISBN文件位于光盘根目录下，记录光盘的ISBN。
- -c<开机文件名称> 制作可开机光盘时，mkisofs会将开机映像文件中的全-eltorito-catalog<开机文件名称>全部内容作成一个文件。
- -C<盘区编号，盘区编号> 将许多节区合成一个映像文件时，必须使用此参数。
- -copyright<版权信息文件> 指定版权信息文件的文件名。
- -d或-omit-period 省略文件后的句号。
- -D或-disable-deep-relocation ISO 9660最多只能处理8层的目录，超过8层的部分，RRIP会自动将它们设置成ISO 9660兼容的格式。使用-D参数可关闭此功能。
- -f或-follow-links 忽略符号连接。
- -h 显示帮助。
- -hide<目录或文件名> 使指定的目录或文件在ISO 9660或Rock RidgeExtensions的系统中隐藏。
- -hide-joliet<目录或文件名> 使指定的目录或文件在Joliet系统中隐藏。
- -J或-joliet 使用Joliet格式的目录与文件名称。
- -l或-full-iso9660-filenames 使用ISO 9660 32字符长度的文件名。
- -L或-allow-leading-dots 允许文件名的第一个字符为句号。
- -log-file<记录文件> 在执行过程中若有错误信息，预设会显示在屏幕上。
- -m<目录或文件名>或-exclude<目录或文件名> 指定的目录或文件名将不会房入映像文件中。
- -M<映像文件>或-prev-session<映像文件> 与指定的映像文件合并。

- -N或-omit-version-number 省略ISO 9660文件中的版本信息。
- -o<映像文件>或-output<映像文件> 指定映像文件的名称。
- -p<数据处理人>或-preparer<数据处理人> 记录光盘的数据处理人。
- -print-size 显示预估的文件系统大小。
- -quiet 执行时不显示任何信息。
- -r或-rational-rock 使用Rock Ridge Extensions，并开放全部文件的读取权限。
- -R或-rock 使用Rock Ridge Extensions。
- -sysid<系统ID> 指定光盘的系统ID。
- -T或-translation-table 建立文件名的转换表，适用于不支持Rock Ridge Extensions的系统上。
- -v或-verbose 执行时显示详细的信息。
- -V<光盘ID>或-volid<光盘ID> 指定光盘的卷册集ID。
- -volset-size<光盘总数> 指定卷册集所包含的光盘张数。
- -volset-seqno<卷册序号> 指定光盘片在卷册集中的编号。
- -x<目录> 指定的目录将不会放入映像文件中。
- -z 建立通透性压缩文件的SUSP记录，此记录目前只在Alpha机器上的Linux有效。

Linux mkswap命令

Linux mkswap命令用于设置交换区(swap area)。

mkswap可将磁盘分区或文件设为Linux的交换区。

语法

```
mkswap [-cf][-v0][-v1][设备名称或文件][交换区大小]
```

参数：

- -c 建立交换区前，先检查是否有损坏的区块。
- -f 在SPARC电脑上建立交换区时，要加上此参数。
- -v0 建立旧式交换区，此为预设值。
- -v1 建立新式交换区。
- [交换区大小] 指定交换区的大小，单位为1024字节。

Linux mpartition命令

Linux mpartition命令用于建立或删除MS-DOS的分区。

mpartition为mtools工具指令，可建立或删除磁盘分区。

语法

```
mpartition [-acdfIprv][-b <磁区数>][-h <磁头数>][l <磁区数>][-s <磁区数>][-t <柱面数>][驱动器代
```

参数：

- -a 将分区设置为可开机分区。
- -b<磁区数> 建立分区时，指定要从第几个磁区开始建立分区。
- -c 建立分区。
- -d 将分区设置为无法开机的分区。
- -f 强制地修改分区而不管检查时发生的错误信息。
- -h<磁头数> 建立分区时，指定分区的磁头数。
- -I 删除全部的分区。
- -l<磁区数> 建立分区时，指定分区的容量大小，单位为磁区数。
- -p 当要重新建立分区时，显示命令列。
- -r 删除分区。
- -s<磁区数> 建立分区时，指定每个磁轨的磁区数。
- -t<柱面数> 建立分区时，指定分区的柱面数。
- -v 与-p参数一并使用，若没有同时下达修改分区的命令，则显示目前分区的状态。

Linux swapon命令

Linux swapon命令用于激活Linux系统中交换空间，Linux系统的内存管理必须使用交换区来建立虚拟内存。

语法

```
/sbin/swapon -a [-v]
/sbin/swapon [-v] [-p priority] specialfile ...
/sbin/swapon [-s]
```

参数说明：

- -h 请帮帮我
- -V 显示版本讯息
- -s 显示简短的装置讯息
- -a 自动启动所有SWAP装置
- -p 设定优先权，你可以在0到32767中间选一个数字给他。或是在 /etc/fstab 里面加上 pri=[value] ([value]就是0~32767中间一个数字)，然后你就可以很方便的直接使用 swapon -a 来启动他们，而且有优先权设定。

swapon 是开启swap.

相对的,便有一个关闭swap的指令,swapoff.

Linux symlinks命令

Linux symlinks命令用于维护符号连接的工具程序。

symlinks可检查目录中的符号连接，并显示符号连接类型。以下为symlinks可判断的符号连接类型：

- absolute：符号连接使用了绝对路径。
- dangling：原始文件已经不存在。
- lengthy：符号连接的路径中包含了多余的"../"。
- messy：符号连接的路径中包含了多余的"/"。
- other_fs：原始文件位于其他文件系统中。
- relative：符号连接使用了相对路径。

语法

```
symlinks [-cdrstv][目录]
```

参数：

- -c 将使用绝对路径的符号连接转换为相对路径。
- -d 移除dangling类型的符号连接。
- -r 检查目录下所有子目录中的符号连接。
- -s 检查lengthy类型的符号连接。
- -t 与-c一并使用时，会显示如何将绝对路径的符号连接转换为相对路径，但不会实际转换。
- -v 显示所有类型的符号连接。

Linux sync命令

Linux sync命令用于数据同步,sync命令是在关闭Linux系统时使用的。

Linux 系统中欲写入硬盘的资料有的时候会了效率起见，会写到 filesystem buffer 中，这个 buffer 是一块记忆体空间，如果欲写入硬盘的资料存于此 buffer 中，而系统又突然断电的话，那么资料就会流失了，sync 指令会将存于 buffer 中的资料强制写入硬盘中。

语法

```
sync
```

Linux mbadblocks命令

Linux mbadblocks命令用于检查MS-DOS文件系统的磁盘是否有损坏的磁区。

mbadblocks为mtools工具指令，可用来扫描MS-DOS文件系统的磁盘驱动器，并标示出损坏的磁区。

语法

```
mbadblocks [驱动器代号]
```

Linux mkfs.minix命令

Linux mkfs.minix命令用于建立Minix文件系统。

mkfs.minix可建立Minix文件系统。

语法

```
mkfs.minix [-cv][-i <inode数目>][-l <文件>][-n <文件名长度>][设备名称][区块数]
```

参数：

- -c 检查是否有损坏的区块。
- -i<inode数目> 指定文件系统的inode总数。
- -l<文件> 从指定的文件中，读取文件系统中损坏区块的信息。
- -n<文件名长度> 指定文件名称长度的上限。
- -v 建立第2版的Minix文件系统。

Linux fsck.ext2命令

Linux fsck.ext2命令用于检查文件系统并尝试修复错误。

当ext2文件系统发生错误时，可用fsck.ext2指令尝试加以修复。

语法

```
fsck.ext2 [-acdfFnprsStvVy][-b <分区第一个磁区地址>][-B <区块大小>][-C <反叙述器>][-I <inode缓冲
```

参数：

- -a 自动修复文件系统，不询问任何问题。
- -b<分区第一个磁区地址> 指定分区的第一个磁区的起始地址，也就是Super Block。
- -B<区块大小> 设置该分区每个区块的大小。
- -c 检查指定的文件系统内，是否存在有损坏的区块。
- -C<反叙述器> 指定反叙述器，fsck.ext2指令会把全部的执行过程，都交由其逆向叙述，便于排错或监控程序执行的情形。
- -d 详细显示指令执行过程，便于排错或分析程序执行的情形。
- -f 强制对该文件系统进行完整检查，纵然该文件系统在慨略检查下没有问题。
- -F 检查文件系统之前，先清理该保存设备块区内的数据。
- -I<inode缓冲区块数> 设置欲检查的文件系统，其inode缓冲区的区块数目。
- -l<损坏区块文件> 把文件中所列出的区块，视为损坏区块并将其标示出来，避免应用程序使用该区块。
- -L<损坏区块文件> 此参数的效果和指定"-l"参数类似，但在参考损坏区块文件标示损坏区块之前，会先将原来标示成损坏区块者统统清楚，即全部重新设置，而非仅是加入新的损坏区块标示。
- -n 把欲检查的文件系统设成只读，并关闭互动模式，否决所有询问的问题。
- -p 此参数的效果和指定"-a"参数相同。
- -P<处理inode大小> 设置fsck.ext2指令所能处理的inode大小为多少。
- -r 此参数将忽略不予处理，仅负责解决兼容性的问题。
- -s 检查文件系统时，交换每对字节的内容。
- -S 此参数的效果和指定"-s"参数类似，但不论该文件系统是否已是标准位顺序，一律交换每对字节的内容。
- -t 显示fsck.ext2指令的时序信息。
- -v 详细显示指令执行过程。
- -V 显示版本信息。
- -y 关闭互动模式，且同意所有询问的问题。

Linux fdisk命令

Linux fdisk是一个创建和维护分区表的程序，它兼容DOS类型的分区表、BSD或者SUN类型的磁盘列表。

语法

```
fdisk [必要参数][选择参数]
```

必要参数：

- -l 列出素所有分区表
- -u 与"-l"搭配使用，显示分区数目

选择参数：

- -s<分区编号> 指定分区
- -v 版本信息

菜单操作说明

- m ：显示菜单和帮助信息
- a ：活动分区标记/引导分区
- d ：删除分区
- l ：显示分区类型
- n ：新建分区
- p ：显示分区信息
- q ：退出不保存
- t ：设置分区号
- v ：进行分区检查
- w ：保存修改
- x ：扩展应用，高级功能

实例

显示当前分区情况：

```
# fdisk -l

Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

  Device Boot   Start     End   Blocks  Id System
/dev/sda1  *      1     13   104391  83 Linux
/dev/sda2       14    1305  10377990  8e Linux LVM

Disk /dev/sdb: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

Disk /dev/sdb doesn't contain a valid partition table
```

显示SCSI硬盘的每个分区情况

```
# fdisk -lu

Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes

  Device Boot   Start     End   Blocks  Id System
/dev/sda1  *     63   208844   104391  83 Linux
/dev/sda2     208845  20964824  10377990  8e Linux LVM

Disk /dev/sdb: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders, total 10485760 sectors
Units = sectors of 1 * 512 = 512 bytes

Disk /dev/sdb doesn't contain a valid partition table
```

Linux losetup命令

Linux losetup命令用于设置循环设备。

循环设备可把文件虚拟成区块设备，籍以模拟整个文件系统，让用户得以将其视为硬盘驱动器，光驱或软驱等设备，并挂入当作目录来使用。

语法

```
losetup [-d][-e <加密方式>][-o <平移数目>][循环设备代号][文件]
```

参数：

- -d 卸除设备。
- -e<加密方式> 启动加密编码。
- -o<平移数目> 设置数据平移的数目。

实例

（1）创建空的磁盘镜像文件，这里创建一个1.44M的软盘

```
$ dd if=/dev/zero of=floppy.img bs=512 count=2880
```

（2）使用 losetup将磁盘镜像文件虚拟成快设备

```
$ losetup /dev/loop1 floppy.img
```

（3）挂载块设备

```
$ mount /dev/loop0 /tmp
```

经过上面的三步之后，我们就可以通过/tmp目录，像访问真实快设备一样来访问磁盘镜像文件floppy.img。

（4） 卸载loop设备

```
$ umount /tmp
$ losetup -d /dev/loop1
```

Linux mkfs命令

使用方式 : mkfs [-V] [-t fstype] [fs-options] filesys [blocks]

Linux mkfs命令用于在特定的分区上建立 linux 文件系统

参数 ：

- device ： 预备检查的硬盘分区，例如：/dev/sda1
- -V : 详细显示模式
- -t : 给定档案系统的型式，Linux 的预设值为 ext2
- -c : 在制做档案系统前，检查该partition 是否有坏轨
- -l bad_blocks_file : 将有坏轨的block资料加到 bad_blocks_file 里面
- block : 给定 block 的大小

实例

在 /dev/hda5 上建一个 msdos 的档案系统，同时检查是否有坏轨存在，并且将过程详细列出来 :

```
mkfs -V -t msdos -c /dev/hda5
```

将sda6分区格式化为ext3格式

```
mfks -t ext3 /dev/sda6
```

注意：这里的文件系统是要指定的，比如 ext3 ；reiserfs ；ext2 ；fat32 ；msdos 等。

Linux getty命令

Linux getty命令用于设置终端机模式，连线速率和管制线路。

getty指令是UNIX之类操作系统启动时所必须的3个步骤之一。

语法

```
getty [-h][-d<组态配置文件>][-r<延迟秒数>][-t<超时秒数>][-w<等待字符串>][终端机编号][连线速率<终端机
```

参数：

- -c<定义配置文件> 指定定义配置文件，预设为/etc/gettydefs。
- -d<组态配置文件> 指定组态配置文件，预设为/etc/conf.getty。
- -h 当传输速率为0时就强制断线。
- -r<延迟秒数> 设置延迟时间。
- -t<超时秒数> 设置等待登入的时间。
- -w<等待字符串> 设置等待回应的字符串。

实例

开启终端：

```
# getty tty7
```

Linux sfdisk命令

Linux sfdisk命令是硬盘分区工具程序。

sfdisk为硬盘分区工具程序，可显示分区的设置信息，并检查分区是否正常。

语法

```
sfdisk [-?Tvx][-d <硬盘>][-g <硬盘>][-l <硬盘>][-s <分区>][-V <硬盘>]
```

参数：

- -?或--help 显示帮助。
- -d<硬盘> 显示硬盘分区的设置。
- -g<硬盘>或--show-geometry<硬盘> 显示硬盘的CHS参数。
- -l<硬盘> 显示后硬盘分区的相关设置。
- -s<分区> 显示分区的大小，单位为区块。
- -T或--list-types 显示所有sfdisk能辨识的文件系统ID。
- -v或--version 显示版本信息。
- -V<硬盘>或--verify<硬盘> 检查硬盘分区是否正常。
- -x或--show-extend 显示扩展分区中的逻辑分区。

实例

显示分区信息：

```
# sfdisk -l

Disk /dev/sda: 1305 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

Device Boot Start End #cyls #blocks Id System
/dev/sda1 * 0+ 12 13- 104391 83 Linux
/dev/sda2 13 1304 1292 10377990 8e Linux LVM
/dev/sda3 0 - 0 0 0 Empty
/dev/sda4 0 - 0 0 0 Empty

Disk /dev/sdb: 652 cylinders, 255 heads, 63 sectors/track

sfdisk: ERROR: sector 0 does not have an msdos signature
/dev/sdb: unrecognized partition
No partitions found
```

Linux swapoff命令

Linux swapoff命令用于关闭系统交换区(swap area)。

swapoff实际上为swapon的符号连接，可用来关闭系统的交换区。

语法

```
swapoff [设备]
```

参数：

- -a 将/etc/fstab文件中所有设置为swap的设备关闭
- -h 帮助信息
- -V 版本信息

实例

显示分区信息:

```
# sfdisk -l //显示分区信息

Disk /dev/sda: 1305 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

  Device Boot Start   End  #cyls  #blocks  Id System
/dev/sda1   *    0+    12    13-  104391  83 Linux
/dev/sda2      13  1304  1292  10377990  8e Linux LVM
/dev/sda3      0     -     0      0  0 Empty
/dev/sda4      0     -     0      0  0 Empty

Disk /dev/sdb: 652 cylinders, 255 heads, 63 sectors/track

sfdisk: ERROR: sector 0 does not have an msdos signature
/dev/sdb: unrecognized partition
No partitions found
```

关闭交换分区。

```
# swapoff /dev/sda2 // 关闭交换分区
```

Linux命令大全 - 网络通讯

apachectl	arpwatch	dip	getty
mingetty	uux	telnet	uulog
uustat	ppp-off	netconfig	nc
httpd	ifconfig	minicom	mesg
dnsconf	wall	netstat	ping
pppstats	samba	setserial	talk
traceroute	tty	newaliases	uuname
netconf	write	statserial	efax
pppsetup	tcpdump	ytalk	cu
smbd	testparm	smbclient	shapecfg

Linux apachectl命令

Linux apachectl命令可用来控制Apache HTTP服务器的程序。

apachectl是slackware内附Apache HTTP服务器的script文件，可供管理员控制服务器，但在其他Linux的Apache HTTP服务器不一定有这个文件。

语法

```
apachectl [configtest][fullstatus][graceful][help][restart][start][status][stop]
```

参数：

- configtest 检查设置文件中的语法是否正确。
- fullstatus 显示服务器完整的状态信息。
- graceful 重新启动Apache服务器，但不会中断原有的连接。
- help 显示帮助信息。
- restart 重新启动Apache服务器。
- start 启动Apache服务器。
- status 显示服务器摘要的状态信息。
- stop 停止Apache服务器。

Linux arpwatch命令

Linux arpwatch命令用于监听网络上ARP的记录。

ARP(Address Resolution Protocol)是用来解析IP与网络装置硬件地址的协议。

arpwatch可监听区域网络中的ARP数据包并记录，同时将监听到的变化通过E-mail来报告。

语法

```
arpwatch [-d][-f<记录文件>][-i<接口>][-r<记录文件>]
```

参数：

- -d 启动排错模式。
- -f<记录文件> 设置存储ARP记录的文件，预设为/var/arpwatch/arp.dat。
- -i<接口> 指定监听ARP的接口，预设的接口为eth0。
- -r<记录文件> 从指定的文件中读取ARP记录，而不是从网络上监听。
- -n 指定附加的本地网络
- -u 指定用户和用户组
- -e 发送邮件给指定用户，非默认的root用户
- -s 指定用户名作为返回地址，而不是默认的用户root

实例

监听网卡eth0的ARP信息

```
arpwatch -i eth0
```

监听ARP的信息，将相关信息记录到相应的文件

```
# arpwatch -i eth0 -f a.log //将信息记录到a.log中
```

Linux nc命令

Linux nc命令用于设置路由器。

执行本指令可设置路由器的相关参数。

语法

```
nc [-hlnruz][-g<网关...>][-G<指向器数目>][-i<延迟秒数>][-o<输出文件>][-p<通信端口>][-s<来源位址>]
```

参数说明：

- -g<网关> 设置路由器跃程通信网关，最丢哦可设置8个。
- -G<指向器数目> 设置来源路由指向器，其数值为4的倍数。
- -h 在线帮助。
- -i<延迟秒数> 设置时间间隔，以便传送信息及扫描通信端口。
- -l 使用监听模式，管控传入的资料。
- -n 直接使用IP地址，而不通过域名服务器。
- -o<输出文件> 指定文件名称，把往来传输的数据以16进制字码倾倒成该文件保存。
- -p<通信端口> 设置本地主机使用的通信端口。
- -r 乱数指定本地与远端主机的通信端口。
- -s<来源位址> 设置本地主机送出数据包的IP地址。
- -u 使用UDP传输协议。
- -v 显示指令执行过程。
- -w<超时秒数> 设置等待连线的时间。
- -z 使用0输入/输出模式，只在扫描通信端口时使用。

实例

TCP端口扫描

```
# nc -v -z -w2 192.168.0.3 1-100
192.168.0.3: inverse host lookup failed: Unknown host
(UNKNOWN) [192.168.0.3] 80 (http) open
(UNKNOWN) [192.168.0.3] 23 (telnet) open
(UNKNOWN) [192.168.0.3] 22 (ssh) open
```

扫描192.168.0.3 的端口 范围是 1-100

扫描UDP端口

```
# nc -u -z -w2 192.168.0.1 1-1000 //扫描192.168.0.3 的端口 范围是 1-1000
```

扫描指定端口

```
# nc -nvv 192.168.0.1 80 //扫描 80端口
(UNKNOWN) [192.168.0.1] 80 (?) open
y  //用户输入
```

Linux dip命令

Linux dip命令用于IP拨号连接。

dip可控制调制解调器，以拨号IP的方式建立对外的双向连接。

语法

```
dip [-aikltv][-m<MTU数目>][-p<协议>][拨号script文件]
```

参数说明：

- -a 询问用户名称与密码。
- -i 启动拨号服务器功能。
- -k 删除执行中的dip程序。
- -l 指定要删除的连线，必须配合-k参数一起使用。
- -m<MTU数目> 设置最大传输单位，预设值为296。
- -p<协议> 设置通信协议。
- -t 进入dip的指令模式。
- -v 执行时显示详细的信息。

实例

建立拨号连接

```
$ dip -t
```

Linux mingetty命令

Linux mingetty命令是精简版的getty。

mingetty适用于本机上的登入程序。

语法

```
mingetty [--long-hostname][--noclear][tty]
```

参数说明：

- --long-hostname 显示完整的主机名称。
- --noclear 在询问登入的用户名称之前不要清楚屏幕画面。

Linux netconfig命令

Linux netconfig命令用于设置网络环境。

这是Slackware发行版内附程序，它具有互动式的问答界面，让用户轻易完成网络环境的设置。

语法

```
netconfig
```

Linux ppp-off命令

Linux ppp命令用于关闭ppp连线。

这是Slackware发行版内附的程序，让用户切断PPP的网络连线。

语法

```
ppp-off
```

实例

关闭ppp连线

```
# ppp-off
```

Linux uustat命令

Linux uustat命令用于显示UUCP目前的状况。

执行uucp与uux指令后，会先将工作送到队列，再由uucico来执行工作。uustat可显示，删除或启动队列中等待执行的工作。

语法

```
uustat [-aeiKmMNpqQRv][-B<行数>][-c<指令>][-C<指令>][-I<配置文件>][-k<工作>][-o<小时>][-r<工作
```

参数说明：

- -a或-all 显示全部的UUCP工作。
- -B<行数>或--mail-lines<行数> 与-M或-N参数一并使用，用来指定邮件中要包含多少行的信息。
- -c<指令>或--command<指令> 显示与<指令>有关的工作。
- -C<指令>或--not-command<指令> 显示与<指令>无关的工作。
- -e或--executions 仅显示待执行的工作。
- -i或--prompt 针对队列中的每项工作，询问使用是否要删除工作。
- -I<配置文件>或--config<配置文件> 指定配置文件。
- -k<工作>或--kill<工作> 删除指定的工作。
- -m或--status 删除全部的工作。
- -M或-mail 将状态信息邮寄给UUCP管理员。
- -N或--notify 将状态信息分别邮寄给提出该项工作的用户。
- -o<小时>或--older-than<小时> 显示超过指定时数的工作。
- -p或--ps 显示负责UUCP锁定的程序。
- -q或--list 显示每台远端主机上所要执行工作的状态。
- -Q或--no-list 不显示工作。
- -r<工作>或--rejuvenate<工作> 重新启动指定的工作。
- -R或--rejuvenate-all 重新启动全部的工作。
- -s<主机>或--system<主机> 显示与<主机>有关的工作。
- -S<主机>或--not-system<主机> 显示与<主机>无关的工作。
- -v或--version 显示版本信息。
- -u<用户>或--user<用户> 显示与<用户>有关的工作。
- -U<用户>或--not-user<用户> 显示与<用户>无关的工作。
- -W<附注>或--comment<附注> 要放在邮件信息中的附注。
- -y<小时>或--younger-than<小时> 显示低于指定时数的工作。

- -x<层级>或--debug<层级> 指定排错层级。
- --help 显示帮助。

实例

显示所有任务

```
# uustat -a
```

显示等待的任务

```
# uustat -e
```

Linux uulog命令

Linux uulog命令用于显示UUCP记录文件。

uulog可用来显示UUCP记录文件中记录。

语法

```
uulog [-DFISv][-<行数>][-f<主机>][-I<配置文件>][-n<行数>][-s<主机>][-u<用户>][-X<层级>][--help
```

参数说明：

- -D或--debuglog 显示排错记录。
- -f<主机>或--follow<主机> 与-F参数类似，但仅显示与指定主机相关的记录。
- -I<配置文件>或--config<配置文件> 指定程序的配置文件。
- -<行数>,-n<行数>或--lines<行数> 显示记录文件中，从最后算起指定行数的数值。
- -s<主机> 仅显示记录文件中，与指定文件相关的记录。
- -S或--statslog 显示统计记录。
- -u<用户>或--suer<用户> 仅显示记录文件中，与指定用户相关的记录。
- -v或--version 显示版本信息。
- -X<层级>或--debug<层级> 设定排错层级。
- --help 显示帮助。

实例

显示uucp log信息

```
# uulog
```

Linux wall命令

Linux wall命令会将讯息传给每一个 mesg 设定为 yes 的上线使用者。当使用终端机介面做为标准传入时, 讯息结束时需加上 EOF (通常用 Ctrl+D)。

使用权限：所有使用者。

语法

```
wall [ message ]
```

实例

传讯息"hi" 给每一个使用者

```
wall hi
```

广播消息

```
# wall Ilove

Broadcast message from root (pts/4) (Thu May 27 16:41:09 2014):

Ilove
```

Linux uux命令

Linux uux命令用于在远端的UUCP主机上执行指令。

uux可在远端的UUCP主机上执行指令或是执行本机上的指令，但在执行时会使用远端电脑的文件。

语法

```
uux [-bcCIjlnrvz][-a<地址>][-g<等级>][-s<文件>][-x<层级>][--help][指令]
```

参数说明：

- -或-p或--stdin 直接从键盘读取要执行的指令。
- -a<地址>或--requestor<地址> 执行邮件地址，以便寄送状态信息。
- -b或--erturn-stdin 在屏幕上显示状态信息。
- -c或--nocopy 不用将文件复制到缓冲区。
- -C或--copy 将文件复制到缓冲区。
- -g<等级>或--grade<等级> 指定文件传送作业的优先顺序。
- -I或--config file 指定uux配置文件。
- -j或--jobid 显示作业编号。
- -l或--link 将本机上的文件连接到缓冲区。
- -n或--notification=no 无论发生任何状态，都不寄邮件通知用户。
- -r或--nouucico 不要立即启动uucico服务程序，仅将作业送到队列中，然后再执行。
- -s<文件>或--status<文件> 将完成状态保存为指定的文件。
- -v或--version 显示版本信息。
- -x<层级>或--debug<层级> 指定排错层级。
- -z或--notification=error 若发生错误，则以邮件来通知用户。
- --help 显示帮助。

实例

在远程主机 uucp 执行命令

```
# uux hnlinux! date  /// 在远程主机 指定date命令查看系统时间
```

Linux telnet命令

Linux telnet命令用于远端登入。

执行telnet指令开启终端机阶段作业，并登入远端主机。

语法

```
telnet [-8acdEfFKLrx][-b<主机别名>][-e<脱离字符>][-k<域名>][-l<用户名称>][-n<记录文件>][-S<服务
```

参数说明：

- -8 允许使用8位字符资料，包括输入与输出。
- -a 尝试自动登入远端系统。
- -b<主机别名> 使用别名指定远端主机名称。
- -c 不读取用户专属目录里的.telnetrc文件。
- -d 启动排错模式。
- -e<脱离字符> 设置脱离字符。
- -E 滤除脱离字符。
- -f 此参数的效果和指定"-F"参数相同。
- -F 使用Kerberos V5认证时，加上此参数可把本地主机的认证数据上传到远端主机。
- -k<域名> 使用Kerberos认证时，加上此参数让远端主机采用指定的领域名，而非该主机的域名。
- -K 不自动登入远端主机。
- -l<用户名称> 指定要登入远端主机的用户名称。
- -L 允许输出8位字符资料。
- -n<记录文件> 指定文件记录相关信息。
- -r 使用类似rlogin指令的用户界面。
- -S<服务类型> 设置telnet连线所需的IP TOS信息。
- -x 假设主机有支持数据加密的功能，就使用它。
- -X<认证形态> 关闭指定的认证形态。

实例

登录远程主机

```
# telnet 192.168.0.5

//登录IP为 192.168.0.5 的远程主机
```

Linux netstat命令

Linux netstat命令用于显示网络状态。

利用netstat指令可让你得知整个Linux系统的网络情况。

语法

```
netstat [-acCeFghilMnNoprstuvVwx][-A<网络类型>][--ip]
```

参数说明：

- -a或--all 显示所有连线中的Socket。
- -A<网络类型>或--<网络类型> 列出该网络类型连线中的相关地址。
- -c或--continuous 持续列出网络状态。
- -C或--cache 显示路由器配置的快取信息。
- -e或--extend 显示网络其他相关信息。
- -F或--fib 显示FIB。
- -g或--groups 显示多重广播功能群组组员名单。
- -h或--help 在线帮助。
- -i或--interfaces 显示网络界面信息表单。
- -l或--listening 显示监控中的服务器的Socket。
- -M或--masquerade 显示伪装的网络连线。
- -n或--numeric 直接使用IP地址，而不通过域名服务器。
- -N或--netlink或--symbolic 显示网络硬件外围设备的符号连接名称。
- -o或--timers 显示计时器。
- -p或--programs 显示正在使用Socket的程序识别码和程序名称。
- -r或--route 显示Routing Table。
- -s或--statistice 显示网络工作信息统计表。
- -t或--tcp 显示TCP传输协议的连线状况。
- -u或--udp 显示UDP传输协议的连线状况。
- -v或--verbose 显示指令执行过程。
- -V或--version 显示版本信息。
- -w或--raw 显示RAW传输协议的连线状况。
- -x或--unix 此参数的效果和指定"-A unix"参数相同。
- --ip或--inet 此参数的效果和指定"-A inet"参数相同。

实例

显示详细的网络状况

```
# netstat -a
```

显示当前户籍UDP连接状况

```
# netstat -nu
```

显示UDP端口号的使用情况

```
# netstat -apu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address        Foreign Address       State    PID/Program name
udp    0   0 *:32768           *:*                   -
udp    0   0 *:nfs            *:*                   -
udp    0   0 *:641            *:*                   3006/rpc.statd
udp    0   0 192.168.0.3:netbios-ns   *:*                   3537/nmbd
udp    0   0 *:netbios-ns        *:*                   3537/nmbd
udp    0   0 192.168.0.3:netbios-dgm   *:*                   3537/nmbd
udp    0   0 *:netbios-dgm        *:*                   3537/nmbd
udp    0   0 *:tftp           *:*                   3346/xinetd
udp    0   0 *:999            *:*                   3366/rpc.rquotad
udp    0   0 *:sunrpc           *:*                   2986/portmap
udp    0   0 *:ipp            *:*                   6938/cupsd
udp    0   0 *:1022           *:*                   3392/rpc.mountd
udp    0   0 *:638            *:*                   3006/rpc.statd
```

显示网卡列表

```
# netstat -i
Kernel Interface table
Iface    MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0    1500  0  181864   0   0   0  141278   0   0   0 BMRU
lo    16436  0   3362   0   0   0   3362   0   0   0 LRU
```

显示组播组的关系

```
# netstat -g
IPv6/IPv4 Group Memberships
Interface    RefCnt Group
--------------- ------ ---------------------
lo        1   ALL-SYSTEMS.MCAST.NET
eth0       1   ALL-SYSTEMS.MCAST.NET
lo        1   ff02::1
eth0       1   ff02::1:ff0a:b0c
eth0       1   ff02::1
```

显示网络统计信息

```
# netstat -s
Ip:
  184695 total packets received
  0 forwarded
  0 incoming packets discarded
```

```
    184687 incoming packets delivered
    143917 requests sent out
    32 outgoing packets dropped
    30 dropped because of missing route
Icmp:
    676 ICMP messages received
    5 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 44
        echo requests: 287
        echo replies: 345
    304 ICMP messages sent
    0 ICMP messages failed
    ICMP output histogram:
        destination unreachable: 17
        echo replies: 287
Tcp:
    473 active connections openings
    28 passive connection openings
    4 failed connection attempts
    11 connection resets received
    1 connections established
    178253 segments received
    137936 segments send out
    29 segments retransmited
    0 bad segments received.
    336 resets sent
Udp:
    5714 packets received
    8 packets to unknown port received.
    0 packet receive errors
    5419 packets sent
TcpExt:
    1 resets received for embryonic SYN_RECV sockets
    ArpFilter: 0
    12 TCP sockets finished time wait in fast timer
    572 delayed acks sent
    3 delayed acks further delayed because of locked socket
    13766 packets directly queued to recvmsg prequeue.
    1101482 packets directly received from backlog
    19599861 packets directly received from prequeue
    46860 packets header predicted
    14541 packets header predicted and directly queued to user
    TCPPureAcks: 12259
    TCPHPAcks: 9119
    TCPRenoRecovery: 0
    TCPSackRecovery: 0
    TCPSACKReneging: 0
    TCPFACKReorder: 0
    TCPSACKReorder: 0
    TCPRenoReorder: 0
    TCPTSReorder: 0
    TCPFullUndo: 0
    TCPPartialUndo: 0
    TCPDSACKUndo: 0
    TCPLossUndo: 0
    TCPLoss: 0
    TCPLostRetransmit: 0
    TCPRenoFailures: 0
    TCPSackFailures: 0
    TCPLossFailures: 0
    TCPFastRetrans: 0
    TCPForwardRetrans: 0
    TCPSlowStartRetrans: 0
    TCPTimeouts: 29
    TCPRenoRecoveryFail: 0
    TCPSackRecoveryFail: 0
    TCPSchedulerFailed: 0
    TCPRcvCollapsed: 0
    TCPDSACKOldSent: 0
    TCPDSACKOfoSent: 0
    TCPDSACKRecv: 0
```

```
  TCPDSACKOfoRecv: 0
  TCPAbortOnSyn: 0
  TCPAbortOnData: 1
  TCPAbortOnClose: 0
  TCPAbortOnMemory: 0
  TCPAbortOnTimeout: 3
  TCPAbortOnLinger: 0
  TCPAbortFailed: 3
  TCPMemoryPressures: 0
```

显示监听的套接口

```
# netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address        Foreign Address       State
tcp    0   0 *:32769           *:*              LISTEN
tcp    0   0 *:nfs            *:*              LISTEN
tcp    0   0 *:644            *:*              LISTEN
tcp    0   0 *:1002           *:*              LISTEN
tcp    0   0 *:netbios-ssn        *:*              LISTEN
tcp    0   0 *:sunrpc          *:*              LISTEN
tcp    0   0 vm-dev:ipp         *:*              LISTEN
tcp    0   0 *:telnet          *:*              LISTEN
tcp    0   0 *:601            *:*              LISTEN
tcp    0   0 *:microsoft-ds       *:*              LISTEN
tcp    0   0 *:http           *:*              LISTEN
tcp    0   0 *:ssh            *:*              LISTEN
tcp    0   0 *:https           *:*              LISTEN
udp    0   0 *:32768           *:*
udp    0   0 *:nfs            *:*
udp    0   0 *:641            *:*
udp    0   0 192.168.0.3:netbios-ns   *:*
udp    0   0 *:netbios-ns        *:*
udp    0   0 192.168.0.3:netbios-dgm   *:*
udp    0   0 *:netbios-dgm        *:*
udp    0   0 *:tftp           *:*
udp    0   0 *:999            *:*
udp    0   0 *:sunrpc          *:*
udp    0   0 *:ipp            *:*
udp    0   0 *:1022           *:*
udp    0   0 *:638            *:*
Active UNIX domain sockets (only servers)
Proto RefCnt Flags    Type    State     I-Node Path
unix 2   [ ACC ]   STREAM   LISTENING   10621 @/tmp/fam-root-
unix 2   [ ACC ]   STREAM   LISTENING   7096  /var/run/acpid.socket
unix 2   [ ACC ]   STREAM   LISTENING   9792  /tmp/.gdm_socket
unix 2   [ ACC ]   STREAM   LISTENING   9927  /tmp/.X11-unix/X0
unix 2   [ ACC ]   STREAM   LISTENING   10489 /tmp/ssh-lbUnUf4552/agent.4552
unix 2   [ ACC ]   STREAM   LISTENING   10558 /tmp/ksocket-root/kdeinit__0
unix 2   [ ACC ]   STREAM   LISTENING   10560 /tmp/ksocket-root/kdeinit-:0
unix 2   [ ACC ]   STREAM   LISTENING   10570 /tmp/.ICE-unix/dcop4664-1270815442
unix 2   [ ACC ]   STREAM   LISTENING   10843 /tmp/.ICE-unix/4735
unix 2   [ ACC ]   STREAM   LISTENING   10591 /tmp/ksocket-root/klauncherah3arc.slave-soc
unix 2   [ ACC ]   STREAM   LISTENING   7763  /var/run/iiim/.iiimp-unix/9010
unix 2   [ ACC ]   STREAM   LISTENING   11047 /tmp/orbit-root/linc-1291-0-1e92c8082411
unix 2   [ ACC ]   STREAM   LISTENING   11053 /tmp/orbit-root/linc-128e-0-dc070659cbb3
unix 2   [ ACC ]   STREAM   LISTENING   8020  /var/run/dbus/system_bus_socket
unix 2   [ ACC ]   STREAM   LISTENING   58927 /tmp/mcop-root/vm-dev-2c28-4beba75f
unix 2   [ ACC ]   STREAM   LISTENING   7860  /tmp/.font-unix/fs7100
unix 2   [ ACC ]   STREAM   LISTENING   7658  /dev/gpmctl
unix 2   [ ACC ]   STREAM   LISTENING   10498 @/tmp/dbus-s2MLJGO5Ci
```

Linux dnsconf命令

Linux dnsconf命令用于设置DNS服务器组态。

dnsconf实际上为linuxconf的符号连接，提供图形截面的操作方式，供管理员管理DNS服务器。

语法

```
dnsconf [--deldomain<域>][--delsecondary<域>][--newdomain<域>][--set<主机><IP>][--setcname<
```

参数说明：

- --deldomain<域> 删除域。
- --delsecondary<域> 删除次级域。
- --newdomain<域> 新增域。
- --set<主机><IP> 新增主机记录。
- --setcname<CNAME><主机> 设置<CNAME>。
- --setmx<域><主机> 指定域的邮件主机。
- --setns<域><主机> 指定域的DNS服务器。
- --unset<主机> 删除DNS中某台主机的记录。

Linux mesg命令

Linux mesg命令用于设置终端机的写入权限。

将mesg设置y时，其他用户可利用write指令将信息直接显示在您的屏幕上。

语法

```
mesg [ny]
```

参数：

- n 不允许气筒用户将信息直接显示在你的屏幕上。
- y 允许气筒用户将信息直接显示在你的屏幕上。

实例

允许其他用户发信息到当前终端。

root 的终端

```
# mesg y //在这个终端设置允许发送消息
```

其他普通用户的终端：

```
$ write root pts/4
hello
hello
EOF //Ctrl+D 结束输入
```

root 的终端 终端显示

```
#
Message from root@w3cschool.cc (as hnlinux) on pts/5 at 14:48 ...
hello
EOF
```

Linux httpd命令

Linux httpd命令是Apache HTTP服务器程序。

httpd为Apache HTTP服务器程序。直接执行程序可启动服务器的服务。

语法

```
httpd [-hlLStvVX][-c<httpd指令>][-C<httpd指令>][-d<服务器根目录>][-D<设定文件参数>][-f<设定文件>
```

参数说明：

- -c<httpd指令> 在读取配置文件前，先执行选项中的指令。
- -C<httpd指令> 在读取配置文件后，再执行选项中的指令。
- -d<服务器根目录> 指定服务器的根目录。
- -D<设定文件参数> 指定要传入配置文件的参数。
- -f<设定文件> 指定配置文件。
- -h 显示帮助。
- -l 显示服务器编译时所包含的模块。
- -L 显示httpd指令的说明。
- -S 显示配置文件中的设定。
- -t 测试配置文件的语法是否正确。
- -v 显示版本信息。
- -V 显示版本信息以及建立环境。
- -X 以单一程序的方式来启动服务器。

实例

检查配置文件语法错误

```
# httpd -t
httpd: Could not determine the server's fully qualified domain name, using 127.0.0.1 for
Syntax OK
```

启动httpd

```
httpd
httpd: Could not determine the server's fully qualified domain name, using 127.0.0.1 for
```

显示编译模块

```
# httpd -l
Compiled in modules:
 core.c
 prefork.c
 http_core.c
 mod_so.c
```

显示配置文件

```
# httpd -L>1.log|tail -n 20 1.log
Maximum number of children alive at the same time
Allowed in *.conf only outside , or
ServerLimit (prefork.c)
Maximum value of MaxClients for this run of Apache
Allowed in *.conf only outside , or
KeepAliveTimeout (http_core.c)
Keep-Alive timeout duration (sec)
Allowed in *.conf only outside , or
MaxKeepAliveRequests (http_core.c)
Maximum number of Keep-Alive requests per connection, or 0 for infinite
Allowed in *.conf only outside , or
KeepAlive (http_core.c)
Whether persistent connections should be On or Off
Allowed in *.conf only outside , or
LoadModule (mod_so.c)
a module name and the name of a shared object file to load it from
Allowed in *.conf only outside , or
LoadFile (mod_so.c)
shared object file or library to load into the server at runtime
Allowed in *.conf only outside , or
```

Linux ifconfig命令

Linux ifconfig命令用于显示或设置网络设备。

ifconfig可设置网络设备的状态，或是显示目前的设置。

语法

```
ifconfig [网络设备][down up -allmulti -arp -promisc][add<地址>][del<地址>][<hw<网络设备类型><
```

参数说明：

- add<地址> 设置网络设备IPv6的IP地址。
- del<地址> 删除网络设备IPv6的IP地址。
- down 关闭指定的网络设备。
- <hw<网络设备类型><硬件地址> 设置网络设备的类型与硬件地址。
- io_addr<I/O地址> 设置网络设备的I/O地址。
- irq<IRQ地址> 设置网络设备的IRQ。
- media<网络媒介类型> 设置网络设备的媒介类型。
- mem_start<内存地址> 设置网络设备在主内存所占用的起始地址。
- metric<数目> 指定在计算数据包的转送次数时，所要加上的数目。
- mtu<字节> 设置网络设备的MTU。
- netmask<子网掩码> 设置网络设备的子网掩码。
- tunnel<地址> 建立IPv4与IPv6之间的隧道通信地址。
- up 启动指定的网络设备。
- -broadcast<地址> 将要送往指定地址的数据包当成广播数据包来处理。
- -pointopoint<地址> 与指定地址的网络设备建立直接连线，此模式具有保密功能。
- -promisc 关闭或启动指定网络设备的promiscuous模式。
- [IP地址] 指定网络设备的IP地址。
- [网络设备] 指定网络设备的名称。

实例

显示网络设备信息

```
# ifconfig
eth0   Link encap:Ethernet HWaddr 00:50:56:0A:0B:0C
     inet addr:192.168.0.3 Bcast:192.168.0.255 Mask:255.255.255.0
     inet6 addr: fe80::250:56ff:fe0a:b0c/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
     RX packets:172220 errors:0 dropped:0 overruns:0 frame:0
     TX packets:132379 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:87101880 (83.0 MiB) TX bytes:41576123 (39.6 MiB)
     Interrupt:185 Base address:0x2024

lo    Link encap:Local Loopback
     inet addr:127.0.0.1 Mask:255.0.0.0
     inet6 addr: ::1/128 Scope:Host
     UP LOOPBACK RUNNING MTU:16436 Metric:1
     RX packets:2022 errors:0 dropped:0 overruns:0 frame:0
     TX packets:2022 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:0
     RX bytes:2459063 (2.3 MiB) TX bytes:2459063 (2.3 MiB)
```

启动关闭指定网卡

```
# ifconfig eth0 down
# ifconfig eth0 up
```

为网卡配置和删除IPv6地址

```
# ifconfig eth0 add 33ffe:3240:800:1005::2/ 64 //为网卡诶之IPv6地址

# ifconfig eth0 del 33ffe:3240:800:1005::2/ 64 //为网卡删除IPv6地址
```

用ifconfig修改MAC地址

```
# ifconfig eth0 down //关闭网卡
# ifconfig eth0 hw ether 00:AA:BB:CC:DD:EE //修改MAC地址
# ifconfig eth0 up //启动网卡
# ifconfig eth1 hw ether 00:1D:1C:1D:1E //关闭网卡并修改MAC地址
# ifconfig eth1 up //启动网卡
```

配置IP地址

```
# ifconfig eth0 192.168.1.56
//给eth0网卡配置IP地址
# ifconfig eth0 192.168.1.56 netmask 255.255.255.0
// 给eth0网卡配置IP地址,并加上子掩码
# ifconfig eth0 192.168.1.56 netmask 255.255.255.0 broadcast 192.168.1.255
// 给eth0网卡配置IP地址,加上子掩码,加上个广播地址
```

启用和关闭ARP协议

```
# ifconfig eth0 arp  //开启
# ifconfig eth0 -arp  //关闭
```

设置最大传输单元

```
# ifconfig eth0 mtu 1500
//设置能通过的最大数据包大小为 1500 bytes
```

Linux minicom命令

Linux minicom命令用于调制解调器通信程序。

minicom是一个相当受欢迎的PPP拨号连线程序。

语法

```
minicom [-8lmMostz][-a<on或0ff>][-c<on或off>][-C<取文件>][-d<编号>][-p<模拟终端机>][-S<script
```

参数说明：

- -8 不要修改任何8位编码的字符。
- -a<on或0ff> 设置终端机属性。
- -c<on或off> 设置彩色模式。
- -C<取文件> 指定取文件，并在启动时开启取功能。
- -d<编号> 启动或直接拨号。
- -l 不会将所有的字符都转成ASCII码。
- -m 以Alt或Meta键作为指令键。
- -M 与-m参数类似。
- -o 不要初始化调制解调器。
- -p <模拟终端机> 使用模拟终端机。
- -s 开启程序设置画面。
- -S<script文件> 在启动时，执行指定的script文件。
- -t 设置终端机的类型。
- -z 在终端机上显示状态列。
- [配置文件] 指定minicom配置文件。

Linux traceroute命令

Linux traceroute命令用于显示数据包到主机间的路径。

traceroute指令让你追踪网络数据包的路由途径，预设数据包大小是40Bytes，用户可另行设置。

语法

```
traceroute [-dFlnrvx][-f<存活数值>][-g<网关>...][-i<网络界面>][-m<存活数值>][-p<通信端口>][-s<来
```

参数说明：

- -d 使用Socket层级的排错功能。
- -f<存活数值> 设置第一个检测数据包的存活数值TTL的大小。
- -F 设置勿离断位。
- -g<网关> 设置来源路由网关，最多可设置8个。
- -i<网络界面> 使用指定的网络界面送出数据包。
- -I 使用ICMP回应取代UDP资料信息。
- -m<存活数值> 设置检测数据包的最大存活数值TTL的大小。
- -n 直接使用IP地址而非主机名称。
- -p<通信端口> 设置UDP传输协议的通信端口。
- -r 忽略普通的Routing Table，直接将数据包送到远端主机上。
- -s<来源地址> 设置本地主机送出数据包的IP地址。
- -t<服务类型> 设置检测数据包的TOS数值。
- -v 详细显示指令的执行过程。
- -w<超时秒数> 设置等待远端主机回报的时间。
- -x 开启或关闭数据包的正确性检验。

实例

显示到达目的地的数据包路由

```
# traceroute www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 66.249.89.99
traceroute to www.l.google.com (66.249.89.99), 30 hops max, 38 byte packets
1 192.168.0.1 (192.168.0.1) 0.653 ms 0.846 ms 0.200 ms
2 118.250.4.1 (118.250.4.1) 36.610 ms 58.438 ms 55.146 ms
3 222.247.28.177 (222.247.28.177) 54.809 ms 39.879 ms 19.186 ms
4 61.187.255.253 (61.187.255.253) 18.033 ms 49.699 ms 72.147 ms
5 61.137.2.177 (61.137.2.177) 32.912 ms 72.947 ms 41.809 ms
6 202.97.46.5 (202.97.46.5) 60.436 ms 25.527 ms 40.023 ms
7 202.97.35.69 (202.97.35.69) 40.049 ms 66.091 ms 44.358 ms
8 202.97.35.110 (202.97.35.110) 42.140 ms 70.913 ms 41.144 ms
9 202.97.35.14 (202.97.35.14) 116.929 ms 57.081 ms 60.336 ms
10 202.97.60.34 (202.97.60.34) 54.871 ms 69.302 ms 64.353 ms
11 * * *
12 209.85.255.80 (209.85.255.80) 95.954 ms 79.844 ms 76.052 ms
   MPLS Label=385825 CoS=5 TTL=1 S=0
13 209.85.249.195 (209.85.249.195) 118.687 ms 120.905 ms 113.936 ms
14 72.14.236.126 (72.14.236.126) 115.843 ms 137.109 ms 186.491 ms
15 nrt04s01-in-f99.1e100.net (66.249.89.99) 168.024 ms 140.551 ms 161.127 ms
```

Linux talk命令

Linux talk命令用于与其他使用者对谈。

使用权限：所有使用者。

语法

```
talk person [ttyname]
```

参数说明：

- person : 预备对谈的使用者帐号，如果该使用者在其他机器上，则可输入 person@machine.name
- ttyname : 如果使用者同时有两个以上的 tty 连线，可以自行选择合适的 tty 传讯息

实例

与现在机器上的使用者Rollaend对谈，此时 Rollaend 只有一个连线

```
talk Rollaend
```

接下来就是等Rollaend回应，若Rollaend接受，则Rollaend输入 `talk jzlee` 即可开始对谈，结束请按 ctrl+c

与linuxfab.cx上的使用者Rollaend对谈，使用pts/2来对谈

```
talk Rollaend@linuxfab.cx pts/2
```

接下来就是等Rollaend回应，若Rollaend接受，则Rollaend输入 `talk jzlee@jzlee.home` 即可开始对谈，结束请按 ctrl+c

注意：若萤幕的字会出现不正常的字元，试著按 ctrl+l 更新萤幕画面。

Linux ping命令

Linux ping命令用于检测主机。

执行ping指令会使用ICMP传输协议，发出要求回应的信息，若远端主机的网络功能没有问题，就会回应该信息，因而得知该主机运作正常。

语法

```
ping [-dfnqrRv][-c<完成次数>][-i<间隔秒数>][-I<网络界面>][-l<前置载入>][-p<范本样式>][-s<数据包大
```

参数说明：

- -d 使用Socket的SO_DEBUG功能。
- -c<完成次数> 设置完成要求回应的次数。
- -f 极限检测。
- -i<间隔秒数> 指定收发信息的间隔时间。
- -I<网络界面> 使用指定的网络界面送出数据包。
- -l<前置载入> 设置在送出要求信息之前，先行发出的数据包。
- -n 只输出数值。
- -p<范本样式> 设置填满数据包的范本样式。
- -q 不显示指令执行过程，开头和结尾的相关信息除外。
- -r 忽略普通的Routing Table，直接将数据包送到远端主机上。
- -R 记录路由过程。
- -s<数据包大小> 设置数据包的大小。
- -t<存活数值> 设置存活数值TTL的大小。
- -v 详细显示指令的执行过程。

实例

检测是否与主机连通

```
# ping www.w3cschool.cc //ping主机
PING aries.m.alikunlun.com (114.80.174.110) 56(84) bytes of data.
64 bytes from 114.80.174.110: icmp_seq=1 ttl=64 time=0.025 ms
64 bytes from 114.80.174.110: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 114.80.174.110: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 114.80.174.110: icmp_seq=4 ttl=64 time=0.034 ms
64 bytes from 114.80.174.110: icmp_seq=5 ttl=64 time=0.028 ms
64 bytes from 114.80.174.110: icmp_seq=6 ttl=64 time=0.028 ms
64 bytes from 114.80.174.110: icmp_seq=7 ttl=64 time=0.034 ms
64 bytes from 114.80.174.110: icmp_seq=8 ttl=64 time=0.034 ms
64 bytes from 114.80.174.110: icmp_seq=9 ttl=64 time=0.036 ms
64 bytes from 114.80.174.110: icmp_seq=10 ttl=64 time=0.041 ms

--- aries.m.alikunlun.com ping statistics ---
10 packets transmitted, 30 received, 0% packet loss, time 29246ms
rtt min/avg/max/mdev = 0.021/0.035/0.078/0.011 ms

//需要手动终止Ctrl+C
```

指定接收包的次数

```
# ping -c 2 www.w3cschool.cc
PING aries.m.alikunlun.com (114.80.174.120) 56(84) bytes of data.
64 bytes from 114.80.174.120: icmp_seq=1 ttl=54 time=6.18 ms
64 bytes from 114.80.174.120: icmp_seq=2 ttl=54 time=15.4 ms

--- aries.m.alikunlun.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1016ms
rtt min/avg/max/mdev = 6.185/10.824/15.464/4.640 ms

//收到两次包后，自动退出
```

多参数使用

```
# ping -i 3 -s 1024 -t 255 g.cn //ping主机
PING g.cn (203.208.37.104) 1024(1052) bytes of data.
1032 bytes from bg-in-f104.1e100.net (203.208.37.104): icmp_seq=0 ttl=243 time=62.5 ms
1032 bytes from bg-in-f104.1e100.net (203.208.37.104): icmp_seq=1 ttl=243 time=63.9 ms
1032 bytes from bg-in-f104.1e100.net (203.208.37.104): icmp_seq=2 ttl=243 time=61.9 ms

--- g.cn ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 61.959/62.843/63.984/0.894 ms, pipe 2
[root@linux ~]#

//-i 3 发送周期为 3秒 -s 设置发送包的大小 -t 设置TTL值为 255
```

Linux pppstats命令

Linux pppstats命令用于显示PPP连线状态。

利用pppstats(point to point protocol status)指令可让你得知PPP连接网络的相关信息。

语法

```
pppstats [-adrv][-c<执行次数>][-w<间隔秒数>][网络界面]
```

参数说明：

- -a 显示绝对统计值。
- -c<执行次数> 设置回报状况的次数。
- -d 显示相对统计值。
- -r 显示数据包压缩比率的统计值。
- -v 显示VJTCP文件头的压缩效率统计值。
- -w<间隔秒数> 设置显示统计信息的间隔时间。

实例

显示ppp的了连接状态

```
# pppstats
```

Linux samba命令

Linux samba命令用于Samba服务器控制。

samba为script文件，可启动，停止Samba服务器或回报目前的状态。

语法

```
samba [start][stop][status][restart]
```

参数说明：

- start 启动Samba服务器的服务。
- stop 停止Samba服务器的服务。
- status 显示Samba服务器目前的状态。
- restart 重新启动Samba服务器。

实例

启动Samba

```
# samba start
```

Linux statserial命令

Linux statserial命令用于显示串口状态。

statserial(status ofserial port)可显示各个接脚的状态，常用来判断串口是否正常。

语法

```
statserial [-dnx][串口设备名称]
```

参数说明：

- -d 以10进制数字来表示串口的状态。
- -n 仅显示一次串口的状态后即结束程序。
- -x 与-n参数类似，但是以16进制来表示。

实例

显示串口状态

```
# statserial /dev/tty1
```

只显示一次串口状态

```
# statserial -n /dev/tty1
```

Linux write命令

Linux write命令用于传讯息给其他使用者。

使用权限：所有使用者。

语法

```
write user [ttyname]
```

参数说明：

- user : 预备传讯息的使用者帐号
- ttyname : 如果使用者同时有两个以上的 tty 连线，可以自行选择合适的 tty 传讯息

实例

传讯息给 Rollaend，此时 Rollaend 只有一个连线

```
write Rollaend
```

接下来就是将讯息打上去，结束请按 ctrl+c

传讯息给 Rollaend，Rollaend 的连线有 pts/2，pts/3

```
write Rollaend pts/2
```

接下来就是将讯息打上去，结束请按 ctrl+c

注意：若对方设定 mesg n，则此时讯席将无法传给对方。

Linux setserial命令

Linux setserial命令用于设置或显示串口的相关信息。

setserial可用来设置串口或显示目前的设置。

语法

```
setserial [-abgGqvVz][设备][串口参数]
```

参数说明：

- -a 显示详细信息。
- -b 显示摘要信息。
- -g 显示串口的相关信息。
- -G 以指令列表的格式来显示信息。
- -q 执行时显示较少的信息。
- -v 执行时显示较多的信息。
- -V 显示版本信息。
- -z 设置前，先将所有的标记归零。

实例

显示串口信息

```
# setserial -g /dev/ttyS2
/dev/ttyS2, UART: unknown, Port: 0x03e8, IRQ: 4
```

Linux tty命令

Linux tty命令用于显示终端机连接标准输入设备的文件名称。

在Linux操作系统中，所有外围设备都有其名称与代号，这些名称代号以特殊文件的类型存放于/dev目录下。你可以执行tty(teletypewriter)指令查询目前使用的终端机的文件名称。

语法

```
tty [-s][--help][--version]
```

参数说明：

- -s或--silent或--quiet 不显示任何信息，只回传状态代码。
- --help 在线帮助。
- --version 显示版本信息。

实例

显示当前终端

```
# tty
/dev/pts/4
```

Linux newaliases命令

Linux newaliases命令会使用一个在 /etc/aliases 中的档案做使用者名称转换的动作。当 sendmail 收到一个要送给 xxx 的信时，它会依据 aliases档的内容送给另一个使用者。这个功能可以创造一个只有在信件系统内才有效的使用者。例如 mailing list 就会用到这个功能，在 mailinglist 中，我们可能会创造一个叫 redlinux@link.ece.uci.edu 的 mailinglist，但实际上并没有一个叫 redlinux 的使用者。实际 aliases 档的内容是将送给这个使用者的信都收给 mailing list 处理程序负责分送的工作。

/etc/aliases 是一个文字模式的档案，sendmail 需要一个二进位格式的 /etc/aliases.db。newaliases 的功能传是将 /etc/aliases 转换成一个 sendmail 所能了解的数据库。

使用权限：系统管理者。

语法

```
newaliases
```

参数说明：没有任何参数。

实例

```
# newaliases
```

下面命令会做相同的事

```
# sendmail -bi
```

Linux uuname命令

Linux uuname命令用于显示全部的UUCP远端主机。

uuname可显示UUCP远端主机。

语法

```
uuname [-alv][-I<配置文件>][--help]
```

参数说明：

- -a或--aliases 显示别名。
- -I<配置文件>或--config<配置文件> 指定程序的配置文件。
- -l或--local 显示本机名称。
- -v或--version 显示版本信息。
- --help 显示帮助。

实例

显示uucp主机名称

```
# uuname
```

Linux netconf命令

Linux netconf命令用于设置各项网络功能。

netconf是Red Hat Linux发行版专门用来调整Linux各项设置的程序。

语法

```
netconf
```

Linux smbd命令

Linux smbd命令用于Samba服务器程序。

smbd为Samba服务器程序，可分享文件与打印机等网络资源供Windows相关的用户端程序存取。

语法

```
smbd [-aDhoP][-d<排错层级>][-i<范围>][-l<记录文件>][-O<连接槽选项>][-p<连接端口编号>][-s<配置文件
```

参数说明：

- -a 所有的连线记录都会加到记录文件中。
- -d<排错层级> 指定记录文件所记载事件的详细程度。
- -D 使用此参数时，smbd会以服务程序的方式在后台执行。
- -h 显示帮助。
- -i<范围> 指定NetBIOS名称的范围。
- -l<记录文件> 指定记录文件的名称。
- -o 每次启动时，会覆盖原有的记录文件。
- -O<连接槽选项> 设置连接槽选项。
- -p<连接端口编号> 设置连接端口编号。
- -P 仅用来测试smbd程序的正确性。
- -s<配置文件> 指定smbd的设置文件。

实例

启动Samba服务器

```
# smbd -D
```

Linux ytalk命令

Linux ytalk命令用于与其他用户交谈。

通过ytalk指令，你可以和其他用户线上交谈，如果想和其他主机的用户交谈，在用户名称后加上其主机名称或IP地址即可。

语法

```
ytalk [-isxY][-h<主机名称IP地址>][用户名称...]
```

参数说明：

- -h<主机名称IP地址> 指定交谈对象所在的远端主机。
- -i 用提醒声响代替显示信息。
- -s 在指令提示符号先开启ytalk交谈窗。
- -x 关闭图形界面。
- -Y 所有必须回应yes或no的问题，都必须用大写英文字母"Y"或"N"回答。

实例

发送消息

```
# who  //显示当前用户
root   :0      Apr 9 20:17
root   pts/1    Apr 9 20:17
w3c   pts/6    May 27 16:47 (192.168.0.1)
root   pts/2    May 27 17:37 (192.168.0.1)
# ytalk w3c //发送消息
hey
```

Linux tcpdump命令

Linux tcpdump命令用于倾倒网络传输数据。

执行tcpdump指令可列出经过指定网络界面的数据包文件头，在Linux操作系统中，你必须是系统管理员。

语法

```
tcpdump [-adeflnNOpqStvx][-c<数据包数目>][-dd][-ddd][-F<表达文件>][-i<网络界面>][-r<数据包文件>
```

参数说明：

- -a 尝试将网络和广播地址转换成名称。
- -c<数据包数目> 收到指定的数据包数目后，就停止进行倾倒操作。
- -d 把编译过的数据包编码转换成可阅读的格式，并倾倒到标准输出。
- -dd 把编译过的数据包编码转换成C语言的格式，并倾倒到标准输出。
- -ddd 把编译过的数据包编码转换成十进制数字的格式，并倾倒到标准输出。
- -e 在每列倾倒资料上显示连接层级的文件头。
- -f 用数字显示网际网络地址。
- -F<表达文件> 指定内含表达方式的文件。
- -i<网络界面> 使用指定的网络截面送出数据包。
- -l 使用标准输出列的缓冲区。
- -n 不把主机的网络地址转换成名字。
- -N 不列出域名。
- -O 不将数据包编码最佳化。
- -p 不让网络界面进入混杂模式。
- -q 快速输出，仅列出少数的传输协议信息。
- -r<数据包文件> 从指定的文件读取数据包数据。
- -s<数据包大小> 设置每个数据包的大小。
- -S 用绝对而非相对数值列出TCP关联数。
- -t 在每列倾倒资料上不显示时间戳记。
- -tt 在每列倾倒资料上显示未经格式化的时间戳记。
- -T<数据包类型> 强制将表达方式所指定的数据包转译成设置的数据包类型。
- -v 详细显示指令执行过程。
- -vv 更详细显示指令执行过程。
- -x 用十六进制字码列出数据包资料。
- -w<数据包文件> 把数据包数据写入指定的文件。

实例

显示TCP包信息

```
# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
23:35:55.129998 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 148872068:148872168(100) ack 418
23:35:55.182357 IP 192.168.0.1.2101 > 192.168.0.3.ssh: . ack 100 win 64240
23:35:55.182397 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 100:200(100) ack 1 win 2100
23:35:55.131713 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 50226+ PTR? 1.0.168.192.in-a
23:35:55.131896 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 50226+ P
23:35:55.154238 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64215: 50226 NX
23:35:55.156298 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: 50226 NXDomain 0/0/0 (42)
23:35:55.159292 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 30304+ PTR? 3.0.168.192.in-a
23:35:55.159449 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 30304+ P
23:35:55.179816 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64215: 30304 NX
23:35:55.181279 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: 30304 NXDomain 0/0/0 (42)
23:35:55.181806 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 200:268(68) ack 1 win 2100
23:35:55.182177 IP 192.168.0.1.2101 > 192.168.0.3.ssh: . ack 268 win 64198
23:35:55.182677 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 43983+ PTR? 112.96.103.202.i
23:35:55.182807 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 43983+ P
23:35:55.183055 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 268:352(84) ack 1 win 2100
23:35:55.201096 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64215: 43983 1/
23:35:55.203087 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: 43983 1/0/0 (72)
23:35:55.204666 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 352:452(100) ack 1 win 2100
23:35:55.204852 IP 192.168.0.1.2101 > 192.168.0.3.ssh: . ack 452 win 64152
23:35:55.205305 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 452:520(68) ack 1 win 2100
23:35:55.205889 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 9318+ PTR? 85.6.250.118.in-a
23:35:55.206071 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 9318+ PT
23:35:55.215338 PPPoE [ses 0x1cb0] IP 115.238.1.45.3724 > 118.250.6.85.64120: P 239275192
23:35:55.216273 IP 115.238.1.45.3724 > 192.168.0.65.2057: P 2392751922:2392751987(65) ack
23:35:55.329204 IP 192.168.0.1.2101 > 192.168.0.3.ssh: . ack 520 win 64135
23:35:55.458214 IP 192.168.0.65.2057 > 115.238.1.45.3724: . ack 65 win 32590
23:35:55.458221 PPPoE [ses 0x1cb0] IP 118.250.6.85.64120 > 115.238.1.45.3724: . ack 65 wi
23:35:55.708228 PPPoE [ses 0x1cb0] IP 115.238.1.45.3724 > 118.250.6.85.64120: P 65:118(53
23:35:55.710213 IP 115.238.1.45.3724 > 192.168.0.65.2057: P 65:118(53) ack 1 win 54
23:35:55.865151 IP 192.168.0.65.2057 > 115.238.1.45.3724: . ack 118 win 32768
23:35:55.865157 PPPoE [ses 0x1cb0] IP 118.250.6.85.64120 > 115.238.1.45.3724: . ack 118 w
23:35:56.242805 IP 192.168.0.65.2057 > 115.238.1.45.3724: P 1:25(24) ack 118 win 32768
23:35:56.242812 PPPoE [ses 0x1cb0] IP 118.250.6.85.64120 > 115.238.1.45.3724: P 1:25(24)
23:35:56.276816 PPPoE [ses 0x1cb0] IP 115.238.1.45.3724 > 118.250.6.85.64120: . ack 25 wi
23:35:56.278240 IP 115.238.1.45.3724 > 192.168.0.65.2057: . ack 25 win 54
23:35:56.349747 PPPoE [ses 0x1cb0] IP 115.238.1.45.3724 > 118.250.6.85.64120: P 118:159(4
23:35:56.351780 IP 115.238.1.45.3724 > 192.168.0.65.2057: P 118:159(41) ack 25 win 54
23:35:56.400051 PPPoE [ses 0x1cb0] IP 119.147.18.44.8000 > 118.250.6.85.4000: UDP, length
23:35:56.475050 IP 192.168.0.65.2057 > 115.238.1.45.3724: . ack 159 win 32762
23:35:56.475063 PPPoE [ses 0x1cb0] IP 118.250.6.85.64120 > 115.238.1.45.3724: . ack 159 w
23:35:56.508968 PPPoE [ses 0x1cb0] IP 115.238.1.45.3724 > 118.250.6.85.64120: P 159:411(2
23:35:56.510182 IP 115.238.1.45.3724 > 192.168.0.65.2057: P 159:411(252) ack 25 win 54
23:35:56.592028 PPPoE [ses 0x1cb0] IP 117.136.2.43.38959 > 118.250.6.85.63283: UDP, lengt

44 packets captured
76 packets received by filter
0 packets dropped by kernel
```

显示指定数量包

```
# tcpdump -c 20
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
23:36:28.949538 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 148875984:148876020(36) ack 4184
23:36:28.994325 IP 192.168.0.1.2101 > 192.168.0.3.ssh: . ack 36 win 64020
23:36:28.994368 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 36:72(36) ack 1 win 2100
23:36:28.950779 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 18242+ PTR? 1.0.168.192.in-a
23:36:28.950948 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 18242+ P
23:36:28.960105 PPPoE [ses 0x1cb0] IP 222.82.119.41.13594 > 118.250.6.85.63283: UDP, leng
23:36:28.962192 IP 222.82.119.41.13594 > 192.168.0.65.13965: UDP, length 36
23:36:28.963118 IP 192.168.0.65.13965 > 222.82.119.41.13594: UDP, length 34
23:36:28.963123 PPPoE [ses 0x1cb0] IP 118.250.6.85.63283 > 222.82.119.41.13594: UDP, leng
23:36:28.970185 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64215: 18242 NX
23:36:28.970413 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: 18242 NXDomain 0/0/0 (42)
23:36:28.972352 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: 17862+ PTR? 3.0.168.192.in-a
23:36:28.972474 PPPoE [ses 0x1cb0] IP 118.250.6.85.64215 > dns2.cs.hn.cn.domain: 17862+ P
23:36:28.982287 PPPoE [ses 0x1cb0] IP 121.12.131.163.13109 > 118.250.6.85.63283: UDP, len
23:36:28.984162 IP 121.12.131.163.13109 > 192.168.0.65.13965: UDP, length 27
23:36:28.985021 IP 192.168.0.65.13965 > 121.12.131.163.13109: UDP, length 103
23:36:28.985027 PPPoE [ses 0x1cb0] IP 118.250.6.85.63283 > 121.12.131.163.13109: UDP, len
23:36:28.991919 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64215: 17862 NX
23:36:28.993142 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: 17862 NXDomain 0/0/0 (42)
23:36:28.993574 IP 192.168.0.3.ssh > 192.168.0.1.2101: P 72:140(68) ack 1 win 2100
20 packets captured
206 packets received by filter
129 packets dropped by kernel
```

精简显示

```
# tcpdump -c 10 -q //精简模式显示 10个包
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
23:43:05.792280 IP 192.168.0.3.ssh > 192.168.0.1.2101: tcp 36
23:43:05.842115 IP 192.168.0.1.2101 > 192.168.0.3.ssh: tcp 0
23:43:05.845074 IP 115.238.1.45.3724 > 192.168.0.65.2057: tcp 0
23:43:05.907155 IP 192.168.0.3.ssh > 192.168.0.1.2101: tcp 36
23:43:05.793880 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: UDP, length 42
23:43:05.794076 PPPoE [ses 0x1cb0] IP 118.250.6.85.64219 > dns2.cs.hn.cn.domain: UDP, len
23:43:05.811127 PPPoE [ses 0x1cb0] IP dns2.cs.hn.cn.domain > 118.250.6.85.64219: UDP, len
23:43:05.814764 IP dns2.cs.hn.cn.domain > 192.168.0.3.32804: UDP, length 42
23:43:05.816404 IP 192.168.0.3.32804 > dns2.cs.hn.cn.domain: UDP, length 42
23:43:05.816545 PPPoE [ses 0x1cb0] IP 118.250.6.85.64219 > dns2.cs.hn.cn.domain: UDP, len
10 packets captured
39 packets received by filter
0 packets dropped by kernel
```

转换克阅读格式

```
# tcpdump -d
(000) ret   #96
```

转换成十进制格式

```
# tcpdump -ddd
1
6 0 0 96
```

Linux cu命令

Linux cu命令用于连接另一个系统主机。

cu(call up)指令可连接另一台主机，并采用类似拨号终端机的接口工作，也可执行简易的文件传输作业。

语法

```
cu [dehnotv][-a<通信端口>][-c<电话号码>][-E<脱离字符>][-I<设置文件>][-l<外围设备代号>][-s<连线速率
```

参数说明：

- -a<通信端口>或-p<通信端口>或--port<通信端口> 使用指定的通信端口进行连线。
- -c<电话号码>或--phone<电话号码> 拨打该电话号码。
- -d 进入排错模式。
- -e或--parity=even 使用双同位检查。
- -E<脱离字符>或--escape<脱离字符> 设置脱离字符。
- -h或--halfduple 使用半双工模式。
- -l<配置文件>或--config<配置文件> 指定要使用的配置文件。
- -l<外围设备代号>或--line<外围设备代号> 指定某项外围设备，作为连接的设备。
- -n或--prompt 拨号时等待用户输入电话号码。
- -o或--parity=odd 使用单同位检查。
- -s<连线速率>或--speed<连线速率>或--baud<连线速率>或-<连线速率> 设置连线的速率，单位以鲍率计算。
- -t或--maper 把CR字符置换成LF+CR字符。
- -v或--version 显示版本信息。
- -x<排错模式>或--debug<排错模式> 使用排错模式。
- -z<系统主机>或--system<系统主机> 连接该系统主机。
- --help 在线帮助。
- --nostop 关闭Xon/Xoff软件流量控制。
- --parity=none 不使用同位检查。

实例

与远程主机连接

```
# cu -c 0102377765
```

Linux efax命令

Linux efax命令用于收发传真。

支持Class 1与Class 2的调制解调器来收发传真。

语法

```
efax [-sw][-a<AT指令>][-c<调制解调器属性>][-d<驱动程序>][-f<字体文件>][-g<指令>][-h<传真标题字符串
```

参数说明：

- -a<AT指令> 以指定的AT指令来接电话。
- -c<调制解调器属性> 设置本机调制解调器的属性。
- -d<驱动程序> 指定调制解调器驱动程序。
- -f<字体文件> 使用指定的字体文件来建立传真标题。
- -g<指令> 若接到的电话为数据，则执行指定的指令。
- -h<传真标题字符串> 指定字符串为每页最前端的标题。
- -i<AT指令> 在调制解调器进入传真模式前，传送AT指令到调制解调器。
- -j<AT指令> 在调制解调器进入传真模式后，传送AT指令到调制解调器。
- -k<AT指令> 在调制解调器离开传真模式前，传送AT指令到调制解调器。
- -l<识别码> 设置本机调制解调器的识别码。
- -o<选项> 使用非标准调制解调器时设置相关选项。
- -q<错误次数> 接收传真时，当每页发生错误次数超过指定的数目时，要求对方重发。
- -r<文件名> 在接收传真时，将每页分别保存成文件。
- -v<信息类型> 选择要印出的信息类型。
- -w 不要接听电话，等待OK或CONNECT的信号。
- -x<UUCP锁定文件> 使用UUCP格式的锁定文件来锁定调制解调器。
- -t<电话号码><传真文件> 以<电话号码>中的号码来拨号，并将<传真文件>传真出去。

Linux pppsetup命令

Linux pppsetup命令用于设置PPP连线。

这是Slackware发行版内附程序，它具有互动式的问答界面，让用户轻易完成PPP的连线设置。

语法

```
pppsetup
```

实例

设置ppp拨号

```
# pppsetup
```

Linux testparm命令

Linux testparm命令用于测试Samba的设置是否正确无误。

执行testparm(test parameter)指令可以简单测试Samba的配置文件，假如测试结果无误，Samba常驻服务就能正确载入该设置值，但并不保证其后的操作如预期般一切正常。

语法

```
testparm [-s][配置文件][<主机名称><IP地址>]
```

参数说明：

- -s 不显示提示符号等待用户按下Enter键，就直接列出Samba服务定义信息。

实例

查看Ssmba配置

```
# testparm
Load smb config files from /etc/samba/smb.conf
Processing section '[homes]'
Processing section '[printers]'
Processing section '[uptech]'
Processing section '[home]'
Loaded services file OK.
Server role: ROLE_STANDALONE
Press enter to see a dump of your service definitions
      ///按下回车继续
# Global parameters
[global]
workgroup = MYGROUP
server string = Samba Server
security = SHARE
encrypt passwords = No
password server = None
log file = /var/log/samba/%m.log
max log size = 50
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
printcap name = /etc/printcap
dns proxy = No
idmap uid = 16777216-33554431
idmap gid = 16777216-33554431
cups options = raw

[homes]
comment = Home Directories
read only = No
browseable = No

[printers]
comment = All Printers
path = /var/spool/samba
printable = Yes
browseable = No

[uptech]
comment = *
path = /home/uptech
read only = No
guest ok = Yes

[home]
comment = *
path = /home
read only = No
guest ok = Yes
```

Linux smbclient命令

Linux smbclient命令可存取SMB/CIFS服务器的用户端程序。

SMB与CIFS为服务器通信协议，常用于Windows95/98/NT等系统。smbclient(samba client)可让Linux系统存取Windows系统所分享的资源。

语法

```
smbclient [网络资源][密码][-EhLN][-B<IP地址>][-d<排错层级>][-i<范围>][-I<IP地址>][-l<记录文件>][
```

参数说明：

- [网络资源] [网络资源]的格式为//服务器名称/资源分享名称。
- [密码] 输入存取网络资源所需的密码。
- -B<IP地址> 传送广播数据包时所用的IP地址。
- -d<排错层级> 指定记录文件所记载事件的详细程度。
- -E 将信息送到标准错误输出设备。
- -h 显示帮助。
- -i<范围> 设置NetBIOS名称范围。
- -I<IP地址> 指定服务器的IP地址。
- -l<记录文件> 指定记录文件的名称。
- -L 显示服务器端所分享出来的所有资源。
- -M<NetBIOS名称> 可利用WinPopup协议，将信息送给选项中所指定的主机。
- -n<NetBIOS名称> 指定用户端所要使用的NetBIOS名称。
- -N 不用询问密码。
- -O<连接槽选项> 设置用户端TCP连接槽的选项。
- -p<TCP连接端口> 指定服务器端TCP连接端口编号。
- -R<名称解析顺序> 设置NetBIOS名称解析的顺序。
- -s<目录> 指定smb.conf所在的目录。
- -t<服务器字码> 设置用何种字符码来解析服务器端的文件名称。
- -T<tar选项> 备份服务器端分享的全部文件，并打包成tar格式的文件。
- -U<用户名称> 指定用户名称。
- -W<工作群组> 指定工作群组名称。

Linux shapecfg命令

Linux shapecfg命令用于管制网络设备的流量。

自Linux-2.15开始，便支持流量管制的功能。

语法

```
shapecfg attach [流量管制器][网络设备]
```

或

```
shapecfg speed [流量管制器][带宽]
```

参数说明：

- attach 将流量管制器与实际的网络设备结合。
- speed 设置流量管制器的对外传输带宽。

Linux命令大全 - 系统管理

adduser	chfn	useradd	date
exit	finger	fwhios	sleep
suspend	groupdel	groupmod	halt
kill	last	lastb	login
logname	logout	ps	nice
procinfo	top	pstree	reboot
rlogin	rsh	sliplogin	screen
shutdown	rwho	sudo	gitps
swatch	tload	logrotate	uname
chsh	userconf	userdel	usermod
vlock	who	whoami	whois
newgrp	renice	su	skill
w	id	free	

Linux date命令

Linux date命令可以用来显示或设定系统的日期与时间，在显示方面，使用者可以设定欲显示的格式，格式设定为一个加号后接数个标记，其中可用的标记列表如下：

时间方面：

- % : 印出 %
- %n : 下一行
- %t : 跳格
- %H : 小时(00..23)
- %I : 小时(01..12)
- %k : 小时(0..23)
- %l : 小时(1..12)
- %M : 分钟(00..59)
- %p : 显示本地 AM 或 PM
- %r : 直接显示时间 (12 小时制，格式为 hh:mm:ss [AP]M)
- %s : 从 1970 年 1 月 1 日 00:00:00 UTC 到目前为止的秒数
- %S : 秒(00..61)
- %T : 直接显示时间 (24 小时制)
- %X : 相当于 %H:%M:%S
- %Z : 显示时区

日期方面：

- %a : 星期几 (Sun..Sat)
- %A : 星期几 (Sunday..Saturday)
- %b : 月份 (Jan..Dec)
- %B : 月份 (January..December)
- %c : 直接显示日期与时间
- %d : 日 (01..31)
- %D : 直接显示日期 (mm/dd/yy)
- %h : 同 %b
- %j : 一年中的第几天 (001..366)
- %m : 月份 (01..12)
- %U : 一年中的第几周 (00..53) (以 Sunday 为一周的第一天的情形)
- %w : 一周中的第几天 (0..6)
- %W : 一年中的第几周 (00..53) (以 Monday 为一周的第一天的情形)
- %x : 直接显示日期 (mm/dd/yy)
- %y : 年份的最后两位数字 (00.99)

- %Y : 完整年份 (0000..9999)

若是不以加号作为开头，则表示要设定时间，而时间格式为 MMDDhhmm[[CC]YY][.ss]，其中 MM 为月份，DD 为日，hh 为小时，mm 为分钟，CC 为年份前两位数字，YY 为年份后两位数字，ss 为秒数。

使用权限：所有使用者。

当您不希望出现无意义的 0 时(比如说 1999/03/07)，则可以在标记中插入 - 符号，比如说 date '+%-H:%-M:%-S' 会把时分秒中无意义的 0 给去掉，像是原本的 08:09:04 会变为 8:9:4。另外，只有取得权限者(比如说 root)才能设定系统时间。

当您以 root 身分更改了系统时间之后，请记得以 clock -w 来将系统时间写入 CMOS 中，这样下次重新开机时系统时间才会持续抱持最新的正确值。

语法

```
date [-u] [-d datestr] [-s datestr] [--utc] [--universal] [--date=datestr] [--set=datestr
```

参数说明：

- -d datestr : 显示 datestr 中所设定的时间 (非系统时间)
- --help : 显示辅助讯息
- -s datestr : 将系统时间设为 datestr 中所设定的时间
- -u : 显示目前的格林威治时间
- --version : 显示版本编号

实例

显示当前时间

```
# date
三 5月 12 14:08:12 CST 2010
# date '+%c'
2010年05月12日 星期三 14时09分02秒
# date '+%D' //显示完整的时间
05/12/10
# date '+%x' //显示数字日期，年份两位数表示
2010年05月12日
# date '+%T' //显示日期，年份用四位数表示
14:09:31
# date '+%X' //显示24小时的格式
14时09分39秒
```

按自己的格式输出

```
# date '+usr_time: $1:%M %P -hey'
usr_time: $1:16 下午 -hey
```

显示时间后跳行，再显示目前日期

```
date '+%T%n%D'
```

显示月份与日数

```
date '+%B %d'
```

显示日期与设定时间(12:34:56)

```
date --date '12:34:56'
```

Linux chfn命令

Linux chfn命令提供使用者更改个人资讯，用于 finger and mail username

使用权限：所有使用者。

语法

```
shell>> chfn
```

实例

改变finger信息

```
# chfn
Changing finger information for root.
Name [root]: hnlinux
Office []: hn
Office Phone []: 888888
Home Phone []: 9999999

Finger information changed.
```

改变账号真实姓名

```
# chfn -f hnunix
Changing finger information for root.
Finger information changed.
```

Linux adduser命令

Linux adduser命令用于新增使用者帐号或更新预设的使用者资料。

adduser 与 useradd 指令为同一指令（经由符号连结 symbolic link）。

使用权限：系统管理员。

adduser是增加使用者。相对的，也有删除使用者的指令，userdel。语法:userdel [login ID]

语法

```
adduser [-c comment] [-d home_dir] [-e expire_date] [-f inactive_time] [-g initial_group]
```

或

```
adduser -D [-g default_group] [-b default_home] [-f default_inactive] [-e default_expire_
```

参数说明：

- -c comment 新使用者位于密码档（通常是 /etc/passwd）的注解资料
- -d home_dir 设定使用者的家目录为 home_dir，预设值为预设的 home 后面加上使用者帐号 loginid
- -e expire_date 设定此帐号的使用期限（格式为 YYYY-MM-DD），预设值为永久有效
- -f inactive_time 范例：

实例

添加一个一般用户

```
# useradd kk //添加用户kk
```

为添加的用户指定相应的用户组

```
# useradd ?g root kk //添加用户kk，并指定用户所在的组为root用户组
```

创建一个系统用户

```
# useradd ?r kk //创建一个系统用户kk
```

为新添加的用户指定/home目录

```
# useradd-d /home/myf kk //新添加用户kk，其home目录为/home/myf
//当用户名kk登录主机时，系统进入的默认目录为/home/myf
```

Linux groupdel命令

Linux groupdel命令用于删除群组。

需要从系统上删除群组时，可用groupdel(group delete)指令来完成这项工作。倘若该群组中仍包括某些用户，则必须先删除这些用户后，方能删除群组。

语法

```
groupdel [群组名称]
```

实例

删除一个群组

```
# groupdel hnuser
```

Linux useradd命令

Linux useradd命令用于建立用户帐号。

useradd可用来建立用户帐号。帐号建好之后，再用passwd设定帐号的密码．而可用userdel删除帐号。使用useradd指令所建立的帐号，实际上是保存在/etc/passwd文本文件中。

语法

```
useradd [-mMnr][-c <备注>][-d <登入目录>][-e <有效期限>][-f <缓冲天数>][-g <群组>][-G <群组>][-
```

或

```
useradd -D [-b][-e <有效期限>][-f <缓冲天数>][-g <群组>][-G <群组>][-s <shell>]
```

参数说明：

- -c<备注>　加上备注文字。备注文字会保存在passwd的备注栏位中。
- -d<登入目录>　指定用户登入时的启始目录。
- -D　变更预设值．
- -e<有效期限>　指定帐号的有效期限。
- -f<缓冲天数>　指定在密码过期后多少天即关闭该帐号。
- -g<群组>　指定用户所属的群组。
- -G<群组>　指定用户所属的附加群组。
- -m　自动建立用户的登入目录。
- -M　不要自动建立用户的登入目录。
- -n　取消建立以用户名称为名的群组．
- -r　建立系统帐号。
- -s<shell>　指定用户登入后所使用的shell。
- -u<uid>　指定用户ID。

实例

添加一般用户

```
# useradd tt
```

为添加的用户指定相应的用户组

```
# useradd -g root tt
```

创建一个系统用户

```
# useradd -r tt
```

为新添加的用户指定home目录

```
# useradd -d /home/myd tt
```

建立用户且制定ID

```
# useradd caojh -u 544
```

Linux groupmod命令

Linux groupmod命令用于更改群组识别码或名称。

需要更改群组的识别码或名称时，可用groupmod指令来完成这项工作。

语法

```
groupmod [-g <群组识别码> <-o>][-n <新群组名称>][群组名称]
```

参数：

- -g <群组识别码> 设置欲使用的群组识别码。
- -o 重复使用群组识别码。
- -n <新群组名称> 设置欲使用的群组名称。

实例

修改组名

```
[root@w3cschool.cc ~]# groupadd linuxso
[root@w3cschool.cc ~]# tail -1 /etc/group
linuxso:x:500:
[root@w3cschool.cc ~]# tail -1 /etc/group
linuxso:x:500:
[root@w3cschool.cc ~]# groupmod -n linux linuxso
[root@w3cschool.cc ~]# tail -1 /etc/group
linux:x:500:
```

Linux logname命令

Linux logname命令用于显示用户名称。

执行logname指令，它会显示目前用户的名称。

语法

```
logname [--help][--version]
```

参数：

- --help 　在线帮助。
- --vesion 　显示版本信息。

实例

显示登录账号的信息：

```
# logname
root
```

Linux logout命令

Linux logout命令用于退出系统。

logout指令让用户退出系统，其功能和login指令相互对应。

语法

```
logout
```

实例

退出系统：

```
[root@w3cschool.cc ~]# logout
```

Linux ps命令

Linux ps命令用于显示当前进程 (process) 的状态。

语法

```
ps [options] [--help]
```

参数：

- ps 的参数非常多, 在此仅列出几个常用的参数并大略介绍含义
- -A 列出所有的行程
- -w 显示加宽可以显示较多的资讯
- -au 显示较详细的资讯
- -aux 显示所有包含其他使用者的行程
- au(x) 输出格式 :
- USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
- USER: 行程拥有者
- PID: pid
- %CPU: 占用的 CPU 使用率
- %MEM: 占用的记忆体使用率
- VSZ: 占用的虚拟记忆体大小
- RSS: 占用的记忆体大小
- TTY: 终端的次要装置号码 (minor device number of tty)
- STAT: 该行程的状态:
- D: 不可中断的静止 (通悸□□缜b进行 I/O 动作)
- R: 正在执行中
- S: 静止状态
- T: 暂停执行
- Z: 不存在但暂时无法消除
- W: 没有足够的记忆体分页可分配
- <: 高优先序的行程
- N: 低优先序的行程
- L: 有记忆体分页分配并锁在记忆体内 (实时系统或捱A I/O)
- START: 行程开始时间
- TIME: 执行的时间
- COMMAND:所执行的指令

实例

```
# ps -A 显示进程信息
PID TTY      TIME CMD
  1 ?    00:00:02 init
  2 ?    00:00:00 kthreadd
  3 ?    00:00:00 migration/0
  4 ?    00:00:00 ksoftirqd/0
  5 ?    00:00:00 watchdog/0
  6 ?    00:00:00 events/0
  7 ?    00:00:00 cpuset
  8 ?    00:00:00 khelper
  9 ?    00:00:00 netns
  10 ?    00:00:00 async/mgr
  11 ?    00:00:00 pm
  12 ?    00:00:00 sync_supers
  13 ?    00:00:00 bdi-default
  14 ?    00:00:00 kintegrityd/0
  15 ?    00:00:02 kblockd/0
  16 ?    00:00:00 kacpid
  17 ?    00:00:00 kacpi_notify
  18 ?    00:00:00 kacpi_hotplug
  19 ?    00:00:27 ata/0
……省略部分结果
30749 pts/0  00:00:15 gedit
30886 ?    00:01:10 qtcreator.bin
30894 ?    00:00:00 qtcreator.bin
31160 ?    00:00:00 dhclient
31211 ?    00:00:00 aptd
31302 ?    00:00:00 sshd
31374 pts/2  00:00:00 bash
31396 pts/2  00:00:00 ps
```

显示指定用户信息

```
# ps -u root //显示root进程用户信息
 PID TTY      TIME CMD
  1 ?    00:00:02 init
  2 ?    00:00:00 kthreadd
  3 ?    00:00:00 migration/0
  4 ?    00:00:00 ksoftirqd/0
  5 ?    00:00:00 watchdog/0
  6 ?    00:00:00 events/0
  7 ?    00:00:00 cpuset
  8 ?    00:00:00 khelper
  9 ?    00:00:00 netns
  10 ?    00:00:00 async/mgr
  11 ?    00:00:00 pm
  12 ?    00:00:00 sync_supers
  13 ?    00:00:00 bdi-default
  14 ?    00:00:00 kintegrityd/0
  15 ?    00:00:02 kblockd/0
  16 ?    00:00:00 kacpid
……省略部分结果
30487 ?    00:00:06 gnome-terminal
30488 ?    00:00:00 gnome-pty-helpe
30489 pts/0  00:00:00 bash
30670 ?    00:00:00 debconf-communi
30749 pts/0  00:00:15 gedit
30886 ?    00:01:10 qtcreator.bin
30894 ?    00:00:00 qtcreator.bin
31160 ?    00:00:00 dhclient
31211 ?    00:00:00 aptd
31302 ?    00:00:00 sshd
31374 pts/2  00:00:00 bash
31397 pts/2  00:00:00 ps
```

显示所有进程信息，连同命令行

```
# ps -ef //显示所有命令，连带命令行
UID    PID PPID C STIME TTY     TIME CMD
root     1   0 0 10:22 ?    00:00:02 /sbin/init
root     2   0 0 10:22 ?    00:00:00 [kthreadd]
root     3   2 0 10:22 ?    00:00:00 [migration/0]
root     4   2 0 10:22 ?    00:00:00 [ksoftirqd/0]
root     5   2 0 10:22 ?    00:00:00 [watchdog/0]
root     6   2 0 10:22 ?    /usr/lib/NetworkManager
……省略部分结果
root   31302 2095 0 17:42 ?    00:00:00 sshd: root@pts/2
root   31374 31302 0 17:42 pts/2  00:00:00 -bash
root   31400   1 0 17:46 ?    00:00:00 /usr/bin/python /usr/sbin/aptd
root   31407 31374 0 17:48 pts/2  00:00:00 ps -ef
```

Linux exit命令

Linux exit命令用于退出目前的shell。

执行exit可使shell以指定的状态值退出。若不设置状态值参数，则shell以预设值退出。状态值0代表执行成功，其他值代表执行失败。exit也可用在script，离开正在执行的script，回到shell。

语法

```
exit [状态值]
```

实例

退出终端

```
# exit
```

Linux finger命令

Linux finger命令可以让使用者查询一些其他使用者的资料。会列出来的资料有：

- Login Name
- User Name
- Home directory
- Shell
- Login status
- mail status
- .plan
- .project
- .forward

其中 .plan、.project 和 .forward 就是使用者在他的 Home Directory 里的 .plan， .project 和 .forward 等档案里的资料。如果没有就没有。finger 指令并不限定于在同一服务器上查询，也可以寻找某一个远端服务器上的使用者。只要给一个像是 E-mail address 一般的地址即可。

使用权限：所有使用者。

语法

```
finger [options] user[@address]
```

参数说明：

- -l 多行显示。
- -s 单行显示。这个选项只显示登入名称、真实姓名、终端机名称、闲置时间、登入时间、办公室号码及电话号码。如果所查询的使用者是远端服务器的使用者，这个选项无效。

实例

列出当前登录用户的相关信息

```
# finger -l //显示用户信息
Login: root Name: root
Directory: /root Shell: /bin/bash
On since Fri Apr 9 20:17 (CST) on :0 (messages off)
On since Fri Apr 9 20:17 (CST) on pts/1 32 days 22 hours idle
On since Fri Apr 9 20:17 (CST) on pts/3 4 hours 5 minutes idle
(messages off)
On since Wed May 12 18:08 (CST) on pts/4 from 192.168.1.10
On since Wed May 12 18:35 (CST) on pts/5 from 192.168.1.10
7 minutes 54 seconds idle
On since Wed May 12 14:37 (CST) on pts/2 from 192.168.1.10
3 hours 14 minutes idle
On since Wed May 12 14:53 (CST) on pts/7 34 minutes 25 seconds idle
(messages off)
On since Wed May 12 16:53 (CST) on pts/8 from 192.168.1.10
30 minutes 18 seconds idle
Mail last read Mon Mar 31 04:02 2008 (CST)
No Plan.
```

显示指定用户信息

```
# finger -m hnlinux
```

显示远程用户信息

```
# finger -m root@192.168.1.13
```

下列指令可以查询本机管理员的资料：

```
finger root
```

其结果如下：

```
Login: root Name: root
Directory: /root Shell: /bin/bash
Never logged in.
No mail.
No Plan.
```

Linux fwhios命令

Linux fwhios命令用于查找并显示用户信息。

本指令的功能有点类似finger指令，它会去查找并显示指定帐号的用户相关信息。不同之处在于fwhois指令是到Network Solutions的WHOIS数据库去查找，该帐号名称必须有在上面注册才能寻获，且名称没有大小写的差别。

语法

```
fwhios [帐号名称]
```

Linux sleep命令

Linux sleep命令可以用来将目前动作延迟一段时间。

使用权限：所有使用者。

语法

```
sleep [--help] [--version] number[smhd]
```

参数说明：

- --help : 显示辅助讯息
- --version : 显示版本编号
- number : 时间长度，后面可接 s、m、h 或 d
- 其中 s 为秒，m 为 分钟，h 为小时，d 为日数

实例

休眠5分钟

```
# sleep 5m
```

显示目前时间后延迟 1 分钟，之后再次显示时间

```
date;sleep 1m;date
```

Linux suspend命令

Linux suspend命令用于暂停执行shell。

suspend为shell内建指令，可暂停目前正在执行的shell。若要恢复，则必须使用SIGCONT信息。

语法

```
suspend [-f]
```

参数说明：

- -f 　若目前执行的shell为登入的shell，则suspend预设无法暂停此shell。若要强迫暂停登入的shell，则必须使用-f参数。

实例

暂停shell

```
# suspend
-bash: suspend: 无法挂起一个登录 shell
# suspend -f
```

Linux login命令

Linux login命令用于登入系统。

login指令让用户登入系统，您亦可通过它的功能随时更换登入身份。在Slackware发行版中，您可在指令后面附加欲登入的用户名称，它会直接询问密码，等待用户输入。当/etc目录里含名称为nologin的文件时，系统只root帐号登入系统，其他用户一律不准登入。

语法

```
login
```

实例

使用新的身份登录系统

```
# login
```

Linux lastb命令

Linux lastb命令用于列出登入系统失败的用户相关信息。

单独执行lastb指令，它会读取位于/var/log目录下，名称为btmp的文件，并把该文件内容记录的登入失败的用户名单，全部显示出来。

语法

```
lastb [-adRx][-f <记录文件>][-n <显示列数>][帐号名称...][终端机编号...]
```

参数说明：

- -a　把从何处登入系统的主机名称或IP地址显示在最后一行。
- -d　将IP地址转换成主机名称。
- -f<记录文件>　指定记录文件。
- -n<显示列数>或-<显示列数>　设置列出名单的显示列数。
- -R　不显示登入系统的主机名称或IP地址。
- -x　显示系统关机，重新开机，以及执行等级的改变等信息。

实例

显示登录失败的用户

```
# lastb
root    tty7      :1         Thu May 13 11:26 - 11:26 (00:00)

btmp begins Thu May 13 11:26:39 2014
```

Linux rlogin命令

Linux rlogin命令用于远端登入。

执行rlogin指令开启终端机阶段操作，并登入远端主机。

语法

```
rlogin [-8EL][-e <脱离字符>][-l <用户名称>][主机名称或IP地址]
```

必要参数：

- -E 忽略escape字符
- -8 只识别8位字的字符
- -L 允许rlogin会话运行在litout模式
- -ec 设置escape字符为c
- -c 断开连接前要求确认
- -a 强制要求远程主机在发送完一个空的本地用户名之后请求一个密码
- -f 向远端主机发送一个本地认证
- -F 向远程主机发送一个可转寄的本地认证
- -7 强制执行7为的传输
- -d 打开用于远端主机通信的TCP套接口的调试
- -k 要求包含远端主机的tisckets
- -x 启动数据传输的DES加密
- -4 只使用 kerkberos的版本4的认证

选择参数：

- -e<字符> 设置退出字符 -l<用户> 指定登陆的用户 -t<终端类型> 设置终端类型

实例

显示rlogin服务是否开启

```
# chkconfig --list //检测rlogin服务是否开启
```

开启rlogin服务

```
# chkconfig rlogin on //开启rlogin服务
```

登陆远程主机

```
# rlogin 192.168.1.88
Password：
Password：
Login incorrect
Login:root
Passwd:
Login incorrect
Login:kk
Passwd:
```

指定用户名登陆远程主机

```
# rlogin 192.168.1.88 -l hnlinux

Passord:
Last login：Mon May 28 15：30:25 from 192.168.1.88

#
```

Linux last命令

Linux last命令用于显示系统开机以来获是从每月初登入者的讯息。

使用权限：所有使用者。

语法

```
shell>> last [options]
```

参数说明：

- -R 省略 hostname 的栏位
- -num 展示前 num 个
- username 展示 username 的登入讯息
- tty 限制登入讯息包含终端机代号

实例

```
shell>> last -R -2
johnney pts/1 Mon Aug 14 20:42 still logged in
johnney pts/0 Mon Aug 14 19:59 still logged in
wtmp begins Tue Aug 1 09:01:10 2000 ### /var/log/wtmp
shell>> last -2 minery
minery pts/0 140.119.217.115 Mon Aug 14 18:37 - 18:40 (00:03)
minery pts/0 140.119.217.115 Mon Aug 14 17:22 - 17:24 (00:02)
wtmp begins Tue Aug 1 09:01:10 2000
```

一般显示方法

```
# last
```

简略显示，并指定显示的个数

```
# last -n 5 -R
root    pts/4    Thu May 13 17:25  still logged in
root    pts/2    Thu May 13 17:23 - 17:25 (00:02)
root    pts/1    Thu May 13 16:46  still logged in
root    pts/7    Thu May 13 15:36  still logged in
root    pts/9    Thu May 13 15:35  still logged in

wtmp begins Thu May 13 18:55:40 2014
```

显示最后一列显示主机IP地址

```
# last -n 5 -a -i
root   pts/4    Thu May 13 17:25  still logged in  192.168.1.10
root   pts/2    Thu May 13 17:23 - 17:25 (00:02)   192.168.1.10
root   pts/1    Thu May 13 16:46  still logged in  192.168.1.10
root   pts/7    Thu May 13 15:36  still logged in  192.168.1.10
root   pts/9    Thu May 13 15:35  still logged in  192.168.1.10

wtmp begins Thu May 13 18:55:40 2014
```

Linux reboot命令

Linux reboot命令用于用来重新启动计算机。

若系统的 runlevel 为 0 或 6 ， 则重新开机，否则以 shutdown 指令（加上 -r 参数）来取代

语法

```
reboot [-n] [-w] [-d] [-f] [-i]
```

参数：

- -n : 在重开机前不做将记忆体资料写回硬盘的动作
- -w : 并不会真的重开机，只是把记录写到 /var/log/wtmp 档案里
- -d : 不把记录写到 /var/log/wtmp 档案里（-n 这个参数包含了 -d）
- -f : 强迫重开机，不呼叫 shutdown 这个指令
- -i : 在重开机之前先把所有网络相关的装置先停止

实例

重新启动

```
# reboot
```

Linux kill命令

Linux kill命令用于删除执行中的程序或工作。

kill可将指定的信息送至程序。预设的信息为SIGTERM(15)，可将指定程序终止。若仍无法终止该程序，可使用SIGKILL(9)信息尝试强制删除程序。程序或工作的编号可利用ps指令或jobs指令查看。

语法

```
kill [-s <信息名称或编号>][程序]　或　kill [-l <信息编号>]
```

参数说明：

- -l <信息编号> 　若不加<信息编号>选项，则-l参数会列出全部的信息名称。
- -s <信息名称或编号> 　指定要送出的信息。
- [程序] 　[程序]可以是程序的PID或是PGID，也可以是工作编号。

实例

杀死进程

```
# kill 12345
```

强制杀死进程

```
# kill -KILL 123456
```

发送SIGHUP信号，可以使用一下信号

```
# kill -HUP pid
```

彻底杀死进程

```
# kill -9 123456
```

显示信号

```
# kill -l
1) SIGHUP     2) SIGINT     3) SIGQUIT     4) SIGILL     5) SIGTRAP
6) SIGABRT     7) SIGBUS     8) SIGFPE     9) SIGKILL    10) SIGUSR1
11) SIGSEGV    12) SIGUSR2    13) SIGPIPE    14) SIGALRM    15) SIGTERM
16) SIGSTKFLT    17) SIGCHLD    18) SIGCONT    19) SIGSTOP    20) SIGTSTP
21) SIGTTIN    22) SIGTTOU    23) SIGURG    24) SIGXCPU    25) SIGXFSZ
26) SIGVTALRM    27) SIGPROF    28) SIGWINCH    29) SIGIO    30) SIGPWR
31) SIGSYS    34) SIGRTMIN    35) SIGRTMIN+1    36) SIGRTMIN+2    37) SIGRTMIN+3
38) SIGRTMIN+4    39) SIGRTMIN+5    40) SIGRTMIN+6    41) SIGRTMIN+7    42) SIGRTMIN+8
43) SIGRTMIN+9    44) SIGRTMIN+10    45) SIGRTMIN+11    46) SIGRTMIN+12    47) SIGRTMIN+1
48) SIGRTMIN+14    49) SIGRTMIN+15    50) SIGRTMAX-14    51) SIGRTMAX-13    52) SIGRTMAX-
53) SIGRTMAX-11    54) SIGRTMAX-10    55) SIGRTMAX-9    56) SIGRTMAX-8    57) SIGRTMAX-7
58) SIGRTMAX-6    59) SIGRTMAX-5    60) SIGRTMAX-4    61) SIGRTMAX-3    62) SIGRTMAX-2
63) SIGRTMAX-1    64) SIGRTMAX
```

杀死指定用户所有进程

```
#kill -9 $(ps -ef | grep hnlinux) //方法一 过滤出hnlinux用户进程
#kill -u hnlinux //方法二
```

Linux halt命令

若系统的 runlevel 为 0 或 6 ，则Linux halt命令关闭系统，否则以 shutdown 指令（加上 -h 参数）来取代。

使用权限：系统管理者。

语法

```
halt [-n] [-w] [-d] [-f] [-i] [-p]
```

参数说明：

- -n : 在关机前不做将记忆体资料写回硬盘的动作
- -w : 并不会真的关机，只是把记录写到 /var/log/wtmp 档案里
- -d : 不把记录写到 /var/log/wtmp 档案里（-n 这个参数包含了 -d） -f : 强迫关机，不呼叫 shutdown 这个指令
- -i : 在关机之前先把所有网络相关的装置先停止
- -p : 当关机的时候，顺便做关闭电源（poweroff）的动作

实例

关闭系统

```
# halt
```

关闭系统并关闭电源

```
# halt -p
```

关闭系统，但不留下纪录

```
# halt -d
```

Linux nice命令

Linux nice命令以更改过的优先序来执行程序，如果未指定程序，则会印出目前的排程优先序，内定的 adjustment 为 10，范围为 -20（最高优先序）到 19（最低优先序）。

使用权限：所有使用者。

语法

```
nice [-n adjustment] [-adjustment] [--adjustment=adjustment] [--help] [--version] [comman
```

参数说明：

- -n adjustment, -adjustment, --adjustment=adjustment 皆为将该原有优先序的增加 adjustment
- --help 显示求助讯息
- --version 显示版本资讯

实例

设置程序运行时的优先级

```
# vi & //后台运行
[1] 15297
# nice vi & //设置默认优先级
[2] 15298

[1]+ Stopped         vi
# nice -n 19 vi & //设置优先级为19
[3] 15299

[2]+ Stopped         nice vi
# nice -n -20 vi & //设置优先级为 -20
[4] 15300

[3]+ Stopped         nice -n 19 vi
# ps -l //显示进程
F S  UID  PID PPID C PRI NI ADDR SZ WCHAN TTY     TIME CMD
4 S   0 15278 15212 0 80  0 - 1208 wait  pts/2  00:00:00 bash
0 T   0 15297 15278 0 80  0 - 2687 signal pts/2  00:00:00 vi
0 T   0 15298 15278 0 90 10 - 2687 signal pts/2  00:00:00 vi
0 T   0 15299 15278 1 99 19 - 2687 signal pts/2  00:00:00 vi
4 T   0 15300 15278 3 60 -20 - 2687 signal pts/2  00:00:00 vi
4 R   0 15301 15278 0 80  0 -  625 -   pts/2  00:00:00 ps

[4]+ Stopped         nice -n -20 vi
```

将 ls 的优先序加 1 并执行

```
nice -n 1 ls
```

将 ls 的优先序加 10 并执行

```
nice ls
```

注意：优先序 (priority) 为操作系统用来决定 CPU 分配的参数，Linux 使用『回合制(round-robin)』的演算法来做 CPU 排程，优先序越高，所可能获得的 CPU时间就越多。

Linux procinfo命令

Linux procinfo命令用于显示系统状态。

procinfo(process information)指令从/proc目录里读取相关数据，将数据妥善整理过后输出到标准输出设备。

语法

```
procinfo [-abdDfhimsSv][-F <输出文件>][-n <间隔秒数>]
```

参数说明：

- -a 显示所有信息。
- -b 显示磁盘设备的区块数目，而非存取数目。
- -d 显示系统信息每秒间的变化差额，而非总和的数值。本参数必须配合"-f"参数使用
- -D 此参数效果和指定"-d"参数类似，但内存和交换文件的信息为总和数值。
- -f 进入全画面的互动式操作界面。
- -F<输出文件> 把信息状态输出到文件保存起来，而非预设的标准输出设备。
- -h 在线帮助。
- -i 显示完整的IRP列表。
- -m 显示系统模块和外围设备等相关信息。
- -n<间隔秒数> 设置全画面互动模式的信息更新速度，单位以秒计算。
- -s 显示系统的内存，磁盘空间，IRP和DMA等信息，此为预设值。
- -S 搭配参数"-d"或"-D"使用时，每秒都会更新信息，不论是否有使用参数"-n"。
- -v 显示版本信息。

实例

显示系统状态

```
# procinfo
```

Linux top命令

Linux top命令用于实时显示 process 的动态。

使用权限：所有使用者。

语法

```
top [-] [d delay] [q] [c] [S] [s] [i] [n] [b]
```

参数说明：

- d : 改变显示的更新速度，或是在交谈式指令列(interactive command)按 s
- q : 没有任何延迟的显示速度，如果使用者是有 superuser 的权限，则 top 将会以最高的优先序执行
- c : 切换显示模式，共有两种模式，一是只显示执行档的名称，另一种是显示完整的路径与名称S : 累积模式，会将己完成或消失的子行程 (dead child process) 的 CPU time 累积起来
- s : 安全模式，将交谈式指令取消, 避免潜在的危机
- i : 不显示任何闲置 (idle) 或无用 (zombie) 的行程
- n : 更新的次数，完成后将会退出 top
- b : 批次档模式，搭配 "n" 参数一起使用，可以用来将 top 的结果输出到档案内

实例

显示进程信息

```
# top
```

显示完整命令

```
# top -c
```

以批处理模式显示程序信息

```
# top -b
```

以累积模式显示程序信息

```
# top -S
```

设置信息更新次数

```
top -n 2

//表示更新两次后终止更新显示
```

设置信息更新时间

```
# top -d 3

//表示更新周期为3秒
```

显示指定的进程信息

```
# top -p 139

//显示进程号为139的进程信息，CPU、内存占用率等
```

显示更新十次后退出

```
top -n 10
```

使用者将不能利用交谈式指令来对行程下命令

```
top -s
```

将更新显示二次的结果输入到名称为 top.log 的档案里

```
top -n 2 -b < top.log
```

Linux pstree命令

Linux pstree命令将所有行程以树状图显示， 树状图将会以 pid (如果有指定) 或是以 init 这个基本行程为根 (root)，如果有指定使用者 id，则树状图会只显示该使用者所拥有的行程。

使用权限：所有使用者。

语法

```
pstree [-a] [-c] [-h|-Hpid] [-l] [-n] [-p] [-u] [-G|-U] [pid|user]
```

或

```
pstree -V
```

参数说明：

- -a 显示该行程的完整指令及参数, 如果是被记忆体置换出去的行程则会加上括号
- -c 如果有重覆的行程名, 则分开列出（预设值是会在前面加上 *）

实例

显示进程的关系

```
pstree
init-+-amd
|-apmd
|-atd
|-httpd---10*[httpd]
%pstree -p
init(1)-+-amd(447)
|-apmd(105)
|-atd(339)
%pstree -c
init-+-amd
|-apmd
|-atd
|-httpd-+-httpd
| |-httpd
| |-httpd
| |-httpd
....
```

特别表明在运行的进程

```
# pstree -apnh //显示进程间的关系
```

同时显示用户名称

```
# pstree -u //显示用户名称
```

Linux shutdown命令

Linux shutdown命令可以用来进行关机程序，并且在关机以前传送讯息给所有使用者正在执行的程序，shutdown 也可以用来重开机。

使用权限：系统管理者。

语法

```
shutdown [-t seconds] [-rkhncfF] time [message]
```

参数说明：

- -t seconds : 设定在几秒钟之后进行关机程序
- -k : 并不会真的关机，只是将警告讯息传送给所有只用者
- -r : 关机后重新开机
- -h : 关机后停机
- -n : 不采用正常程序来关机，用强迫的方式杀掉所有执行中的程序后自行关机
- -c : 取消目前已经进行中的关机动作
- -f : 关机时，不做 fcsk 动作(检查 Linux 档系统)
- -F : 关机时，强迫进行 fsck 动作
- time : 设定关机的时间
- message : 传送给所有使用者的警告讯息

实例

立即关机

```
# shutdown -h now
```

指定5分钟后关机

```
# shutdown +5 “System will shutdown after 5 minutes” //5分钟够关机并显示警告信息
```

Linux screen命令

Linux screen命令用于多重视窗管理程序。

screen为多重视窗管理程序。此处所谓的视窗，是指一个全屏幕的文字模式画面。通常只有在使用telnet登入主机或是使用老式的终端机时，才有可能用到screen程序。

语法

```
screen [-AmRvx -ls -wipe][-d <作业名称>][-h <行数>][-r <作业名称>][-s <shell>][-S <作业名称>]
```

参数说明：

- -A 　将所有的视窗都调整为目前终端机的大小。
- -d<作业名称> 　将指定的screen作业离线。
- -h<行数> 　指定视窗的缓冲区行数。
- -m 　即使目前已在作业中的screen作业，仍强制建立新的screen作业。
- -r<作业名称> 　恢复离线的screen作业。
- -R 　先试图恢复离线的作业。若找不到离线的作业，即建立新的screen作业。
- -s<shell> 　指定建立新视窗时，所要执行的shell。
- -S<作业名称> 　指定screen作业的名称。
- -v 　显示版本信息。
- -x 　恢复之前离线的screen作业。
- -ls或--list 　显示目前所有的screen作业。
- -wipe 　检查目前所有的screen作业，并删除已经无法使用的screen作业。

实例

创建 screen 终端

```
# screen //创建 screen 终端
```

创建 screen 终端 并执行任务

```
# screen vi ~/main.c //创建 screen 终端 ，并执行 vi命令
```

离开 screen 终端

```
# screen vi ~/main.c //创建 screen 终端 ，并执行 vi命令

#include

main ()
{

}

"~/mail.c"       0,0-1

在 screen 终端 下 按下 Ctrl+a d键
```

重新连接离开的 screen 终端

```
# screen -ls  //显示已创建的screen终端
There are screens on:
2433.pts-3.linux    (2013年10月20日 16时48分59秒)    (Detached)
2428.pts-3.linux    (2013年10月20日 16时48分05秒)    (Detached)
2284.pts-3.linux    (2013年10月20日 16时14分55秒)    (Detached)
2276.pts-3.linux    (2013年10月20日 16时13分18秒)    (Detached)
4 Sockets in /var/run/screen/S-root.

# screen -r 2276 //连接 screen_id 为 2276 的 screen终端
```

Linux sliplogin命令

Linux sliplogin命令用于将SLIP接口加入标准输入。

sliplogin可将SLIP接口加入标准输入，把一般终端机的连线变成SLIP连线。通常可用来建立SLIP服务器，让远端电脑以SLIP连线到服务器。sliplogin活去检查/etc/slip/slip.hosts文件中是否有相同的用户名称。通过检查后，sliplogin会调用执行shell script来设置IP地址，子网掩码等网络界面环境。此shell script通常是/etc/slip/slip.login。

语法

```
sliplogin [用户名称]
```

实例

改变用户的连接方式

```
# sliplogin kk // 改变用户的连接方式
```

Linux rsh命令

Linux rsh命令用于远端登入的Shell。

rsh(remote shell)提供用户环境，也就是Shell，以便指令能够在指定的远端主机上执行。

语法

```
rsh [-dn][-l <用户名称>][主机名称或IP地址][执行指令]
```

参数说明：

- -d　使用Socket层级的排错功能。
- -l<用户名称>　指定要登入远端主机的用户名称。
- -n　把输入的指令号向代号为/dev/null的特殊外围设备。

实例

开启rsh服务

```
# chkconfig --list //检测rlogin服务是否开启

# chkconfig rsh on //开启rsh服务

# chkconfig -list //检测开启的服务
```

远程命令执行

```
# rsh -l hnlinux 192.168.1.88 /bin/ls //远程执行ls命令
```

Linux rwho命令

Linux rwho命令用于查看系统用户。

rwho指令的效果类似who指令，但它会显示局域网里所有主机的用户。主机必须提供rwhod常驻服务的功能，方可使用rwho指令。

语法

```
rwho [-a]
```

参数说明：

- -a　列出所有的用户，包括闲置时间超过1个小时以上的用户。

实例

显示本地局域网内的所有用户

```
# rwho
root   snail-hnlinux:pts/2 May 14 17:42
```

Linux sudo命令

Linux sudo命令以系统管理者的身份执行指令，也就是说，经由 sudo 所执行的指令就好像是 root 亲自执行。

使用权限：在 /etc/sudoers 中有出现的使用者。

语法

```
sudo -V
```

```
sudo -h
```

```
sudo -l
```

```
sudo -v
```

```
sudo -k
```

```
sudo -s
```

```
sudo -H
```

```
sudo [ -b ] [ -p prompt ] [ -u username/#uid] -s
```

```
sudo command
```

参数说明：

- -V 显示版本编号
- -h 会显示版本编号及指令的使用方式说明
- -l 显示出自己（执行 sudo 的使用者）的权限
- -v 因为 sudo 在第一次执行时或是在 N 分钟内没有执行（N 预设为五）会问密码，这个参数是重新做一次确认，如果超过 N 分钟，也会问密码
- -k 将会强迫使用者在下一次执行 sudo 时问密码（不论有没有超过 N 分钟）
- -b 将要执行的指令放在背景执行

- -p prompt 可以更改问密码的提示语，其中 %u 会代换为使用者的帐号名称， %h 会显示主机名称
- -u username/#uid 不加此参数，代表要以 root 的身份执行指令，而加了此参数，可以以 username 的身份执行指令（#uid 为该 username 的使用者号码）
- -s 执行环境变数中的 SHELL 所指定的 shell ，或是 /etc/passwd 里所指定的 shell
- -H 将环境变数中的 HOME （家目录）指定为要变更身份的使用者家目录（如不加 -u 参数就是系统管理者 root ）
- command 要以系统管理者身份（或以 -u 更改为其他人）执行的指令

实例

sudo命令使用

```
$ sudo ls
[sudo] password for hnlinux:
hnlinux is not in the sudoers file. This incident will be reported.
```

指定用户执行命令

```
# sudo -u userb ls -l
```

显示sudo设置

```
$ sudo -L //显示sudo设置
Available options in a sudoers ``Defaults'' line:

syslog: Syslog facility if syslog is being used for logging
syslog_goodpri: Syslog priority to use when user authenticates successfully
syslog_badpri: Syslog priority to use when user authenticates unsuccessfully
long_otp_prompt: Put OTP prompt on its own line
ignore_dot: Ignore '.' in $PATH
mail_always: Always send mail when sudo is run
mail_badpass: Send mail if user authentication fails
mail_no_user: Send mail if the user is not in sudoers
mail_no_host: Send mail if the user is not in sudoers for this host
mail_no_perms: Send mail if the user is not allowed to run a command
tty_tickets: Use a separate timestamp for each user/tty combo
lecture: Lecture user the first time they run sudo
lecture_file: File containing the sudo lecture
authenticate: Require users to authenticate by default
root_sudo: Root may run sudo
log_host: Log the hostname in the (non-syslog) log file
log_year: Log the year in the (non-syslog) log file
shell_noargs: If sudo is invoked with no arguments, start a shell
set_home: Set $HOME to the target user when starting a shell with -s
always_set_home: Always set $HOME to the target user's home directory
path_info: Allow some information gathering to give useful error messages
fqdn: Require fully-qualified hostnames in the sudoers file
insults: Insult the user when they enter an incorrect password
requiretty: Only allow the user to run sudo if they have a tty
env_editor: Visudo will honor the EDITOR environment variable
rootpw: Prompt for root's password, not the users's
runaspw: Prompt for the runas_default user's password, not the users's
targetpw: Prompt for the target user's password, not the users's
use_loginclass: Apply defaults in the target user's login class if there is one
set_logname: Set the LOGNAME and USER environment variables
```

```
stay_setuid: Only set the effective uid to the target user, not the real uid
preserve_groups: Don't initialize the group vector to that of the target user
loglinelen: Length at which to wrap log file lines (0 for no wrap)
timestamp_timeout: Authentication timestamp timeout
passwd_timeout: Password prompt timeout
passwd_tries: Number of tries to enter a password
umask: Umask to use or 0777 to use user's
logfile: Path to log file
mailerpath: Path to mail program
mailerflags: Flags for mail program
mailto: Address to send mail to
mailfrom: Address to send mail from
mailsub: Subject line for mail messages
badpass_message: Incorrect password message
timestampdir: Path to authentication timestamp dir
timestampowner: Owner of the authentication timestamp dir
exempt_group: Users in this group are exempt from password and PATH requirements
passprompt: Default password prompt
passprompt_override: If set, passprompt will override system prompt in all cases.
runas_default: Default user to run commands as
secure_path: Value to override user's $PATH with
editor: Path to the editor for use by visudo
listpw: When to require a password for 'list' pseudocommand
verifypw: When to require a password for 'verify' pseudocommand
noexec: Preload the dummy exec functions contained in 'noexec_file'
noexec_file: File containing dummy exec functions
ignore_local_sudoers: If LDAP directory is up, do we ignore local sudoers file
closefrom: File descriptors >= %d will be closed before executing a command
closefrom_override: If set, users may override the value of `closefrom' with the -C optio
setenv: Allow users to set arbitrary environment variables
env_reset: Reset the environment to a default set of variables
env_check: Environment variables to check for sanity
env_delete: Environment variables to remove
env_keep: Environment variables to preserve
role: SELinux role to use in the new security context
type: SELinux type to use in the new security context
askpass: Path to the askpass helper program
env_file: Path to the sudo-specific environment file
sudoers_locale: Locale to use while parsing sudoers
visiblepw: Allow sudo to prompt for a password even if it would be visisble
pwfeedback: Provide visual feedback at the password prompt when there is user input
fast_glob: Use faster globbing that is less accurate but does not access the filesystem
umask_override: The umask specified in sudoers will override the user's, even if it is mo
```

以root权限执行上一条命令

```
$ sudo !!
```

以特定用户身份进行编辑文本

```
$ sudo -u uggc vi ~www/index.html
//以 uggc 用户身份编辑  home 目录下www目录中的 index.html 文件
```

列出目前的权限

```
sudo -l
```

列出 sudo 的版本资讯

```
sudo -V
```

Linux gitps命令

Linux gitps命令用于报告程序状况。

gitps(gnu interactive tools process status)是用来报告并管理程序执行的指令，基本上它就是通过ps指令来报告，管理程序，也能通过gitps指令随时中断，删除不必要的程序。因为gitps指令会去执行ps指令，所以其参数和ps指令相当类似。

语法

```
gitps [acefgjlnrsSTuvwxX][p <程序识别码>][t <终端机编号>][U <帐号名称>]
```

参数说明：

- a 显示 现行终端机下的所有程序，包括其他用户的程序。
- c 列出程序时，显示每个程序真正的指令名称，而不包含路径，参数或是常驻服务的标示.
- e 列出程序时，显示每个程序所使用的环境变量。
- f 用ASCII字符显示树状结构，表达程序间的相互关系。
- g 显示现行终端机下的所有程序，包括群组领导者的程序。
- j 采用工作控制的格式来显示程序状况。
- l 采用纤细的格式来显示程序状况。
- n 以数字来表示USER和WCHAN栏位。
- p<程序识别码> 指定程序识别码，并列出该程序的状况。
- r 只列出现行终端机正在执行中的程序。
- s 采用程序信号的格式显示程序状况。
- S 列出程序时，包括已中断的子程序信息。
- t<终端机机标号> 指定终端机编号，并列出属于该终端机的程序的状况。
- T 显示现行终端机下的所有程序。
- u 以用户为主的格式来显示程序状况。
- U<帐号名称> 列出属于该用户的程序的状况。
- v 采用虚拟内存的格式显示程序状况。
- w 采用宽阔的格式来显示程序状况。
- x 显示所有程序，不以终端机来区分。
- X 采用旧试的Linux i386登陆格式显示程序状况。

实例

显示指定用户信息

```
# gitps hnlinux
```

Linux uname命令

Linux uname命令用于显示系统信息。

uname可显示电脑以及操作系统的相关信息。

语法

```
uname [-amnrsv][--help][--version]
```

参数说明：

- -a或--all 显示全部的信息。
- -m或--machine 显示电脑类型。
- -n或-nodename 显示在网络上的主机名称。
- -r或--release 显示操作系统的发行编号。
- -s或--sysname 显示操作系统名称。
- -v 显示操作系统的版本。
- --help 显示帮助。
- --version 显示版本信息。

实例

显示系统信息

```
# uname -a
Linux snail-hnlinux 2.6.32-21-generic #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010 i686 GN
```

显示计算机类型

```
# uname -m
i686
```

显示计算机名

```
# uname -n
snail-hnlinux
```

显示操作系统发行编号

```
# uname -r
2.6.32-21-generic
```

显示操作系统名称

```
# uname -s
Linux
```

显示系统时间

```
# uname -v
#32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2014
```

Linux logrotate命令

Linux logrotate命令用于管理记录文件。

使用logrotate指令，可让你轻松管理系统所产生的记录文件。它提供自动替换，压缩，删除和邮寄记录文件，每个记录文件都可被设置成每日，每周或每月处理，也能在文件太大时立即处理。您必须自行编辑，指定配置文件，预设的配置文件存放在/etc目录下，文件名称为logrotate.conf。

语法

```
logrotate [-?dfv][-s <状态文件>][--usage][配置文件]
```

参数说明：

- -?或--help 　在线帮助。
- -d或--debug 　详细显示指令执行过程，便于排错或了解程序执行的情况。
- -f或--force 　强行启动记录文件维护操作，纵使logrotate指令认为没有需要亦然。
- -s<状态文件>或--state=<状态文件> 　使用指定的状态文件。
- -v或--version 　显示指令执行过程。
- -usage 　显示指令基本用法。

实例

指定记录文件

```
# logrotate /root/log.config
```

Linux tload命令

Linux tload命令用于显示系统负载状况。

tload指令使用ASCII字符简单地以文字模式显示系统负载状态。假设不给予终端机编号，则会在执行tload指令的终端机显示负载情形。

语法

```
tload [-V][-d <间隔秒数>][-s <刻度大小>][终端机编号]
```

参数说明：

- -d<间隔秒数> 设置tload检测系统负载的间隔时间，单位以秒计算。
- -s<刻度大小> 设置图表的垂直刻度大小，单位以列计算。
- -V 显示版本信息。

实例

显示系统负载

```
# tload
```

Linux swatch命令

Linux swatch命令用于系统监控程序。

swatch可用来监控系统记录文件，并在发现特定的事件时，执行指定的动作。swatch所监控的事件以及对应事件的动作都存放在swatch的配置文件中。预设的配置文件为拥护根目录下的.swatchrc。然而在Red Hat Linux的预设用户根目录下并没有.swatchrc配置文件，您可将/usr/doc/swatch-2.2/config_files/swatchrc.personal文件复制到用户根目录下的.swatchrc，然后修改.swatchrc所要监控的事件及执行的动作。

语法

```
swatch [-A <分隔字符>][-c <设置文件>][-f <记录文件>][-I <分隔字符>][-P <分隔字符>][-r <时间>][-t
```

参数说明：

- -A<分隔字符> 预设配置文件中，动作的分隔字符，预设为逗号。
- -c<设置文件> 指定配置文件，而不使用预设的配置文件。
- -f<记录文件> 检查指定的记录文件，检查完毕后不会继续监控该记录文件。
- -I<分隔字符> 指定输入记录的分隔字符，预设为换行字符。
- -P<分隔字符> 指定配置文件中，事件的分隔字符，预设为逗号。
- -r<时间> 在指定的时间重新启动。
- -t<记录文件> 检查指定的记录文件，并且会监控加入记录文件中的后继记录。

实例

开启系统监视

```
# swatch
```

Linux chsh命令

Linux chsh命令用于更改使用者 shell 设定。

使用权限：所有使用者。

语法

```
shell>> chsh
```

实例

```
shell>> chsh
Changing fihanging shell for user1
Password: [del]
New shell [/bin/tcsh]: ### [是目前使用的 shell]
[del]
shell>> chsh -l ### 展示 /etc/shells 档案内容
/bin/bash
/bin/sh
/bin/ash
/bin/bsh
/bin/tcsh
/bin/csh
```

改变当前的shell。当前的shell 设置为//bin/bash，通过chsh命令，改变shell的设置/bin/csh。

```
# chsh
Changing shell for root.
New shell [/bin/bash]: /bin/csh //输入新的shell地址
Shell changed.
```

通过 -s 参数改变当前的shell设置

```
# chsh -s /bin/csh //改变当前设置为 /bin/csh
Changing shell for root.
Shell not changed.
```

Linux whoami命令

Linux whoami命令用于显示自身用户名称。

显示自身的用户名称，本指令相当于执行"id -un"指令。

语法

```
whoami [--help][--version]
```

参数说明：

- --help 　在线帮助。
- --version 　显示版本信息。

实例

显示用户名

```
# whoami
root
```

Linux who命令

Linux who命令用于显示系统中有哪些使用者正在上面，显示的资料包含了使用者 ID、使用的终端机、从哪边连上来的、上线时间、呆滞时间、CPU 使用量、动作等等。

使用权限：所有使用者都可使用。

语法

```
who - [husfV] [user]
```

参数说明：

- -h : 不要显示标题列
- -u : 不要显示使用者的动作/工作
- -s : 使用简短的格式来显示
- -f : 不要显示使用者的上线位置
- -V : 显示程序版本

实例

显示当前登录系统的用户

```
# who  //显示当前登录系统的用户
root   tty7     2014-05-13 12:12 (:0)
root   pts/0    2014-05-14 17:09 (:0.0)
root   pts/1    2014-05-14 18:51 (192.168.1.17)
root   pts/2    2014-05-14 19:48 (192.168.1.17)
```

显示标题栏

```
# who -H
NAME   LINE     TIME        COMMENT
root   tty7     2014-05-13 12:12 (:0)
root   pts/0    2014-05-14 17:09 (:0.0)
root   pts/1    2014-05-14 18:51 (192.168.1.17)
root   pts/2    2014-05-14 19:48 (192.168.1.17)
```

显示用户登录来源

```
# who -l -H
NAME   LINE     TIME        IDLE     PID COMMENT
LOGIN  tty4     2014-05-13 12:11         852 id=4
LOGIN  tty5     2014-05-13 12:11         855 id=5
LOGIN  tty2     2014-05-13 12:11         862 id=2
LOGIN  tty3     2014-05-13 12:11         864 id=3
LOGIN  tty6     2014-05-13 12:11         867 id=6
LOGIN  tty1     2014-05-13 12:11        1021 id=1
```

显示终端属性

```
# who -T -H
NAME    LINE     TIME        COMMENT
root   + tty7     2014-05-13 12:12 (:0)
root   + pts/0    2014-05-14 17:09 (:0.0)
root   - pts/1    2014-05-14 18:51 (192.168.1.17)
root   - pts/2    2014-05-14 19:48 (192.168.1.17)
```

只显示当前用户

```
# who -m -H
NAME   LINE     TIME        COMMENT
root   pts/1    2014-05-14 18:51 (192.168.1.17)
```

精简模式显示

```
# who -q
root root root root
# users=4
```

Linux vlock命令

Linux vlock命令用于锁住虚拟终端。

执行vlock(virtual console lock)指令可锁住虚拟终端，避免他人使用。

语法

```
vlock [-achv]
```

参数说明：

- -a或--all 锁住所有的终端阶段作业，如果您在全屏幕的终端中使用本参数，则会将用键盘
- 切换终端机的功能一并关闭。
- -c或--current 锁住目前的终端阶段作业，此为预设值。
- -h或--help 在线帮助。
- -v或--version 显示版本信息。

实例

锁定虚拟终端

```
# vlock
```

Linux usermod命令

Linux usermod命令用于修改用户帐号。

usermod可用来修改用户帐号的各项设定。

语法

```
usermod [-LU][-c <备注>][-d <登入目录>][-e <有效期限>][-f <缓冲天数>][-g <群组>][-G <群组>][-l
```

参数说明：

- -c<备注>　修改用户帐号的备注文字。
- -d登入目录>　修改用户登入时的目录。
- -e<有效期限>　修改帐号的有效期限。
- -f<缓冲天数>　修改在密码过期后多少天即关闭该帐号。
- -g<群组>　修改用户所属的群组。
- -G<群组>　修改用户所属的附加群组。
- -l<帐号名称>　修改用户帐号名称。
- -L　锁定用户密码，使密码无效。
- -s<shell>　修改用户登入后所使用的shell。
- -u<uid>　修改用户ID。
- -U　解除密码锁定。

实例

更改登录目录

```
# usermod -d /home/hnlinux root
```

改变用户的uid

```
# usermod -u 777 root
```

Linux userdel命令

Linux userdel命令用于删除用户帐号。

userdel可删除用户帐号与相关的文件。若不加参数，则仅删除用户帐号，而不删除相关文件。

语法

```
userdel [-r][用户帐号]
```

参数说明：

- -r 　删除用户登入目录以及目录中所有文件。

实例

删除用户账号

```
# userdel hnlinux
```

Linux userconf命令

Linux userconf命令用于用户帐号设置程序。

userconf实际上为linuxconf的符号连接，提供图形界面的操作方式，供管理员建立与管理各类帐号。若不加任何参数，即进入图形界面。

语法

```
userconf [--addgroup <群组>][--adduser <用户ID><群组><用户名称><shell>][--delgroup <群组>][--
```

参数说明：

- --addgroup<群组> 新增群组。
- --adduser<用户ID><群组><用户名称><shell> 新增用户帐号。
- --delgroup<群组> 删除群组。
- --deluser<用户ID> 删除用户帐号。
- --help 显示帮助。

实例

新增用户

```
# userconf --adduser 666 tt lord /bin/bash //新增用户账号
```

Linux id命令

Linux id命令用于显示用户的ID，以及所属群组的ID。

id会显示用户以及所属群组的实际与有效ID。若两个ID相同，则仅显示实际ID。若仅指定用户名称，则显示目前用户的ID。

语法

```
id [-gGnru][--help][--version][用户名称]
```

参数说明：

- -g或--group　显示用户所属群组的ID。
- -G或--groups　显示用户所属附加群组的ID。
- -n或--name　显示用户，所属群组或附加群组的名称。
- -r或--real　显示实际ID。
- -u或--user　显示用户ID。
- -help　显示帮助。
- -version　显示版本信息。

实例

显示当前用户信息

```
# id //显示当前用户ID
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel) c
```

显示用户群组的ID

```
# id -g
0
```

显示所有群组的ID

```
# id -g
0 1 2 3 4 5 6 10
```

显示指定用户信息

```
# id hnlinux
```

Linux w命令

Linux w命令用于显示目前登入系统的用户信息。

执行这项指令可得知目前登入系统的用户有哪些人，以及他们正在执行的程序。

单独执行 w 指令会显示所有的用户，您也可指定用户名称，仅显示某位用户的相关信息。

语法

```
w [-fhlsuV][用户名称]
```

参数说明：

- -f　开启或关闭显示用户从何处登入系统。
- -h　不显示各栏位的标题信息列。
- -l　使用详细格式列表，此为预设值。
- -s　使用简洁格式列表，不显示用户登入时间，终端机阶段作业和程序所耗费的CPU时间。
- -u　忽略执行程序的名称，以及该程序耗费CPU时间的信息。
- -V　显示版本信息。

实例

显示当前用户

```
w  //显示当前用户，不显示登录位置
19:50:14 up 9:27, 4 users, load average: 0.31, 0.26, 0.18
USER   TTY   FROM       LOGIN@  IDLE  JCPU  PCPU WHAT
root   tty7   :0        Thu12  31:39m 10:10  0.60s gnome-session
root   pts/0  :0.0       17:09  2:18m 15.26s 0.15s bash
root   pts/1  192.168.1.17   18:51  1.00s 1.24s 0.14s -bash
root   pts/2  192.168.1.17   19:48  60.00s 0.05s 0.05s -bash
```

不显示登录位置

```
w -f
19:53:59 up 9:31, 4 users, load average: 0.05, 0.16, 0.15
USER   TTY    LOGIN@  IDLE  JCPU  PCPU WHAT
root   tty7   Thu12  31:43m 10:10  0.60s gnome-session
root   pts/0   17:09  2:21m 15.26s 0.15s bash
root   pts/1   18:51  0.00s 1.04s 0.14s -bash
root   pts/2   19:48  4:45  0.05s 0.05s -bash
```

以精简模式显示

```
w -s
19:54:37 up 9:31, 4 users, load average: 0.24, 0.19, 0.16
USER   TTY   FROM         IDLE WHAT
root   tty7   :0         31:43m gnome-session
root   pts/0  :0.0        2:22m bash
root   pts/1  192.168.1.17   0.00s -bash
root   pts/2  192.168.1.17   5:23 -bash
```

不显示标题

```
w -h
root   tty7   :0         Thu12  31:44m 10:10  0.60s gnome-session
root   pts/0  :0.0        17:09  2:23m 15.26s 0.15s bash
root   pts/1  192.168.1.17   18:51  0.00s 1.05s 0.14s -bash
root   pts/2  192.168.1.17   19:48  5:54  0.05s 0.05s -bash
```

Linux skill命令

Linux skill命令送个讯号给正在执行的程序，预设的讯息为 TERM (中断)，较常使用的讯息为 HUP、INT、KILL、STOP、CONT 和 0。

讯息有三种写法：分别为 -9、-SIGKILL、-KILL，可以使用 -l 或 -L 已列出可使用的讯息。

使用权限：所有使用者。

其他相关的命令：kill

语法

```
skill [signal to send] [options] 选择程序的规则
```

一般参数：

- -f 快速模式/尚未完成
- -i 互动模式/ 每个动作将要被确认
- -v 详细输出/ 列出所选择程序的资讯
- -w 智能警告讯息/ 尚未完成
- -n 没有动作/ 显示程序代号

参数：选择程序的规则可以是：终端机代号、使用者名称、程序代号、命令名称。

- -t 终端机代号 (tty 或 pty)
- -u 使用者名称
- -p 程序代号 (pid)
- -c 命令名称可使用的讯号

以下列出已知的讯号名称、讯号代号、功能。

名称（代号）	功能/描述
ALRM 14	离开
HUP 1	离开
INT 2	离开
KILL 9	离开/强迫关闭
PIPE 13	离开
POLL	离开
PROF	离开

TERM 15	离开
USR1	离开
USR2	离开
VTALRM	离开
STKFLT	离开/只适用于i386、m68k、arm 和 ppc 硬件
UNUSED	离开/只适用于i386、m68k、arm 和 ppc 硬件
TSTP	停止/产生与内容相关的行为
TTIN	停止/产生与内容相关的行为
TTOU	停止/产生与内容相关的行为
STOP	停止/强迫关闭
CONT	重新启动/如果在停止状态则重新启动，否则忽略
PWR	忽略/在某些系统中会离开
WINCH	忽略
CHLD	忽略
ABRT 6	核心
FPE 8	核心
ILL 4	核心
QUIT 3	核心
SEGV 11	核心
TRAP 5	核心
SYS	核心/或许尚未实作
EMT	核心/或许尚未实作
BUS	核心/核心失败
XCPU	核心/核心失败
XFSZ	核心/核心失败

实例

停止所有在 PTY 装置上的程序

```
skill -KILL -v pts/*
```

停止三个使用者 user1、user2、user3

```
skill -STOP user1 user2 user3
```

Linux su命令

Linux su命令用于变更为其他使用者的身份，除 root 外，需要键入该使用者的密码。

使用权限：所有使用者。

语法

```
su [-fmp] [-c command] [-s shell] [--help] [--version] [-] [USER [ARG]]
```

参数说明：

- -f 或 --fast 不必读启动档（如 csh.cshrc 等），仅用于 csh 或 tcsh
- -m -p 或 --preserve-environment 执行 su 时不改变环境变数
- -c command 或 --command=command 变更为帐号为 USER 的使用者并执行指令（command）后再变回原来使用者
- -s shell 或 --shell=shell 指定要执行的 shell （bash csh tcsh 等），预设值为 /etc/passwd 内的该使用者（USER） shell
- --help 显示说明文件
- --version 显示版本资讯
-
    - -l 或 --login 这个参数加了之后，就好像是重新 login 为该使用者一样，大部份环境变数（HOME SHELL USER等等）都是以该使用者（USER）为主，并且工作目录也会改变，如果没有指定 USER ，内定是 root
- USER 欲变更的使用者帐号
- ARG 传入新的 shell 参数

实例

变更帐号为 root 并在执行 ls 指令后退出变回原使用者

```
su -c ls root
```

变更帐号为 root 并传入 -f 参数给新执行的 shell

```
su root -f
```

变更帐号为 clsung 并改变工作目录至 clsung 的家目录（home dir）

```
su - clsung
```

切换用户

```
hnlinux@w3cschool.cc:~$ whoami //显示当前用户
hnlinux
hnlinux@w3cschool.cc:~$ pwd //显示当前目录
/home/hnlinux
hnlinux@w3cschool.cc:~$ su root //切换到root用户
密码：
root@w3cschool.cc:/home/hnlinux# whoami
root
root@w3cschool.cc:/home/hnlinux# pwd
/home/hnlinux
```

切换用户，改变环境变量

```
hnlinux@w3cschool.cc:~$ whoami //显示当前用户
hnlinux
hnlinux@w3cschool.cc:~$ pwd //显示当前目录
/home/hnlinux
hnlinux@w3cschool.cc:~$ su - root //切换到root用户
密码：
root@w3cschool.cc:/home/hnlinux# whoami
root
root@w3cschool.cc:/home/hnlinux# pwd //显示当前目录
/root
```

Linux renice命令

Linux renice命令用于重新指定一个或多个行程（Process）的优先序（一个或多个将根据参数而定）。

注意：每一个行程（Process）都有一个唯一的（unique）id。

使用权限：所有使用者。

语法

```
renice priority [[-p] pid ...] [[-g] pgrp ...] [[-u] user ...]
```

参数说明：

- -p pid 重新指定行程的 id 为 pid 的行程的优先序
- -g pgrp 重新指定行程群组(process group)的 id 为 pgrp 的行程 (一个或多个) 的优先序
- -u user 重新指定行程拥有者为 user 的行程的优先序

实例

将行程 id 为 987 及 32 的行程与行程拥有者为 daemon 及 root 的优先序号码加 1

```
renice +1 987 -u daemon root -p 32
```

Linux newgrp命令

Linux newgrp命令用于登入另一个群组。

newgrp指令类似login指令，当它是以相同的帐号，另一个群组名称，再次登入系统。欲使用newgrp指令切换群组，您必须是该群组的用户，否则将无法登入指定的群组。单一用户要同时隶属多个群组，需利用交替用户的设置。若不指定群组名称，则newgrp指令会登入该用户名称的预设群组。

语法

```
newgrp [群组名称]
```

实例

改变群组

```
# newgrp root
```

Linux whois命令

Linux whois命令用于查找并显示用户信息。

whois指令会去查找并显示指定帐号的用户相关信息，因为它是到Network Solutions的WHOIS数据库去查找，所以该帐号名称必须在上面注册方能寻获，且名称没有大小写的差别。

语法

```
whois [帐号名称]
```

实例

显示指定用户信息

```
# whois root

//查找root用户信息
```

查询域名描述信息

```
# whois .Lx138.COm

Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

...省略部分内容
```

查询域名信息

```
# whois Lx138.COm

The Registry database contains ONLY .COM, .NET, .EDU domains and
Registrars.
Domain Name ..................... Lx138.COm
Name Server ..................... dns15.hichina.com
                 dns16.hichina.com
Registrant ID ................... hc937242545-cn

...省略部分内容
```

查询域名信息省略法律声明

```
# whois -H Lx138.COm

...省略内容
```

指定端口查询

```
# whois -p 80 Lx138.COm

...省略内容
```

Linux free命令

Linux free命令用于显示内存状态。

free指令会显示内存的使用情况，包括实体内存，虚拟的交换文件内存，共享内存区段，以及系统核心使用的缓冲区等。

语法

```
free [-bkmotV][-s <间隔秒数>]
```

参数说明：

- -b 　以Byte为单位显示内存使用情况。
- -k 　以KB为单位显示内存使用情况。
- -m 　以MB为单位显示内存使用情况。
- -o 　不显示缓冲区调节列。
- -s<间隔秒数> 　持续观察内存使用状况。
- -t 　显示内存总和列。
- -V 　显示版本信息。

实例

显示内存使用情况

```
# free //显示内存使用信息
total used free shared buffers cached
Mem: 254772 184568 70204 0 5692 89892
-/+ buffers/cache: 88984 165788
Swap: 524280 65116 459164
```

以总和的形式显示内存的使用信息

```
# free -t //以总和的形式查询内存的使用信息
total used free shared buffers cached
Mem: 254772 184868 69904 0 5936 89908
-/+ buffers/cache: 89024 165748
Swap: 524280 65116 459164
Total: 779052 249984 529068
```

周期性的查询内存使用信息

```
# free -s 10 //每10s 执行一次命令
total used free shared buffers cached
Mem: 254772 187628 67144 0 6140 89964
-/+ buffers/cache: 91524 163248
Swap: 524280 65116 459164

total used free shared buffers cached
Mem: 254772 187748 67024 0 6164 89940
-/+ buffers/cache: 91644 163128
Swap: 524280 65116 459164
```

Linux命令大全 - 系统设定

reset	clear	alias	dircolors
aumix	bind	chroot	clock
crontab	declare	depmod	dmesg
enable	eval	export	pwunconv
grpconv	rpm	insmod	kbdconfig
lilo	liloconfig	lsmod	minfo
set	modprobe	ntsysv	mouseconfig
passwd	pwconv	rdate	resize
rmmod	grpunconv	modinfo	time
setup	sndconfig	setenv	setconsole
timeconfig	ulimit	unset	chkconfig
apmd	hwclock	mkkickstart	fbset
unalias	SVGATextMode		

Linux bind命令

Linux bind命令用于显示或设置键盘按键与其相关的功能。

您可以利用bind命令了解有哪些按键组合与其功能，也可以自行指定要用哪些按键组合。

语法

```
bind [-dlv][-f <按键配置文件>][-m <按键配置>][-q <功能>]
```

参数说明：

- -d 　显示按键配置的内容。
- -f<按键配置文件> 　载入指定的按键配置文件。
- -l 　列出所有的功能。
- -m<按键配置> 　指定按键配置。
- -q<功能> 　显示指定功能的按键。
- -v 　列出目前的按键配置与其功能。

实例

显示按键组合的所有功能

```
# bind -l //显示按键组合的内容
abort
accept-line
alias-expand-line
arrow-key-prefix
backward-byte
backward-char
backward-delete-char
backward-kill-line
backward-kill-word
backward-word
beginning-of-history
beginning-of-line
……省略部分内容
vi-goto-mark
vi-insert-beg
vi-insertion-mode
vi-match
vi-movement-mode
vi-next-word
vi-overstrike
vi-overstrike-delete
vi-prev-word
vi-put
vi-redo
vi-replace
vi-rubout
vi-search
vi-search-again
vi-set-mark
vi-subst
vi-tilde-expand
vi-yank-arg
vi-yank-to
yank
yank-last-arg
yank-nth-arg
yank-pop
```

显示当前按键组合的设置

```
# bind -l
abort
accept-line
alias-expand-line
arrow-key-prefix
backward-byte
backward-char
backward-delete-char
backward-kill-line
backward-kill-word
backward-word
beginning-of-history
beginning-of-line
call-last-kbd-macro
capitalize-word
character-search
character-search-backward
clear-screen
complete
complete-command
complete-filename
complete-hostname
complete-into-braces
complete-username
complete-variable
copy-backward-word
copy-forward-word
```

```
copy-region-as-kill
dabbrev-expand
delete-char
delete-char-or-list
delete-horizontal-space
digit-argument
display-shell-version
do-lowercase-version
downcase-word
dump-functions
dump-macros
dump-variables
dynamic-complete-history
edit-and-execute-command
emacs-editing-mode
end-kbd-macro
end-of-history
end-of-line
exchange-point-and-mark
forward-backward-delete-char
forward-byte
forward-char
forward-search-history
forward-word
glob-complete-word
glob-expand-word
glob-list-expansions
history-and-alias-expand-line
history-expand-line
history-search-backward
history-search-forward
insert-comment
insert-completions
insert-last-argument
kill-line
kill-region
kill-whole-line
kill-word
magic-space
menu-complete
menu-complete-backward
next-history
non-incremental-forward-search-history
non-incremental-forward-search-history-again
non-incremental-reverse-search-history
non-incremental-reverse-search-history-again
old-menu-complete
operate-and-get-next
overwrite-mode
possible-command-completions
possible-completions
possible-filename-completions
possible-hostname-completions
possible-username-completions
possible-variable-completions
previous-history
quoted-insert
redraw-current-line
re-read-init-file
reverse-search-history
revert-line
self-insert
set-mark
shell-backward-kill-word
shell-backward-word
shell-expand-line
shell-forward-word
shell-kill-word
skip-csi-sequence
start-kbd-macro
tab-insert
tilde-expand
```

```
transpose-chars
transpose-words
tty-status
undo
universal-argument
unix-filename-rubout
unix-line-discard
unix-word-rubout
upcase-word
vi-append-eol
vi-append-mode
vi-arg-digit
vi-back-to-indent
vi-bword
vi-bWord
vi-change-case
vi-change-char
vi-change-to
vi-char-search
vi-column
vi-complete
vi-delete
vi-delete-to
vi-editing-mode
vi-end-word
vi-eof-maybe
vi-eword
vi-eWord
vi-fetch-history
vi-first-print
vi-fword
vi-fWord
vi-goto-mark
vi-insert-beg
vi-insertion-mode
vi-match
vi-movement-mode
vi-next-word
vi-overstrike
vi-overstrike-delete
vi-prev-word
vi-put
vi-redo
vi-replace
vi-rubout
vi-search
vi-search-again
vi-set-mark
vi-subst
vi-tilde-expand
vi-yank-arg
vi-yank-to
yank
yank-last-arg
yank-nth-arg
yank-pop
root@snail-hnlinux:~#
root@snail-hnlinux:~#
root@snail-hnlinux:~#
root@snail-hnlinux:~#
root@snail-hnlinux:~# bind -v
set bind-tty-special-chars on
set blink-matching-paren on
set byte-oriented off
set completion-ignore-case off
set convert-meta off
set disable-completion off
set echo-control-characters on
set enable-keypad off
set enable-meta-key on
set expand-tilde off
set history-preserve-point off
```

```
set horizontal-scroll-mode off
set input-meta on
set mark-directories on
set mark-modified-lines off
set mark-symlinked-directories off
set match-hidden-files on
set meta-flag on
set output-meta on
set page-completions on
set prefer-visible-bell on
set print-completions-horizontally off
set revert-all-at-newline off
set show-all-if-ambiguous off
set show-all-if-unmodified off
set skip-completed-text off
set visible-stats off
set bell-style audible
set comment-begin #
set completion-prefix-display-length 0
set completion-query-items 100
set editing-mode emacs
set history-size 1000
set keymap emacs
```

列出指定功能的按键和按键组合

```
# bind -q abort
//请用 调用abort “C-g”, “C-xC-g”, “eC-g”.

# bind -q accept-line //列出功能“accept-line”按键以及组合按键
//请用 调用accept-line “C-j”, “C-m”.
```

Linux aumix命令

Linux aumix命令用于设置音效装置。

aumix(audio mixer)命令设置各项音效装置的信号强度以及指定播放与录音的装置。

语法

```
aumix [-123bcilmoprstvwWx][(+/-)强度][PqR][-dfhILqS]
```

参数说明：[-123bcilmoprstvwWx]为频道参数，用来指定装置的频道；[PqR]可用来指定播放或录音装置；[-dfhILqS] 则为指令参数。若不加任何参数，aumix会显示简单的图形界面供调整设置频道参数。

- -1 　输入信号线 1 。
- -2 　输入信号线 2 。
- -3 　输入信号线 3 。
- -b 　低音。
- -c 　CD。
- -i 　输入信号强度。
- -m 　麦克风。
- -o 　输出信号强度。
- -p 　PC喇叭。
- -r 　录音。
- -s 　合成器。
- -t 　高音。
- -v 　主音量。
- -w 　PCM。
- -W 　PCM2。
- -x 　混音器。
- (+/-)强度 　出现(+/-)时，代表在原有的强度上加减指定值。若未使用(+/-)，则直接将强度设为指定值。 　指定音效装置
- P 　指定播放装置。
- q 　显示频道设置。
- R 　指定录音装置。

指令参数：

- -d 　指定音效装置的名称。
- -f 　指定存储或载入设置的文件。

- -h　在使用时显示信息。
- -I　以图形界面方式来执行aumix。
- -L　从$HOME/.aumixrc或/etc/aumixrc载入设置。
- -q　显示所有频道的设置值。
- -S　将设置值保存至/HOME/.aumixrc。

实例

设置音效设备

```
# aumix
```

Linux dircolors命令

Linux dircolors命令用于设置 ls 指令在显示目录或文件时所用的色彩。

dircolors可根据[色彩配置文件]来设置LS_COLORS环境变量或是显示设置LS_COLORS环境变量的shell指令。

语法

```
dircolors [色彩配置文件]
```

或

```
dircolors [-bcp][--help][--version]
```

参数说明：

- -b或--sh或--bourne-shell 显示在Boume shell中，将LS_COLORS设为目前预设置的shell指令。
- -c或--csh或--c-shell 显示在C shell中，将LS_COLORS设为目前预设置的shell指令。
- -p或--print-database 显示预设置
- -help 显示帮助。
- -version 显示版本信息。

实例

显示默认值

```
# dircolors -p //显示默认值
# Configuration file for dircolors, a utility to help you set the
# LS_COLORS environment variable used by GNU ls with the --color option.
# Copyright (C) 1996, 1999-2008
# Free Software Foundation, Inc.
# Copying and distribution of this file, with or without modification,
# are permitted provided the copyright notice and this notice are preserved.
# The keywords COLOR, OPTIONS, and EIGHTBIT (honored by the
# slackware version of dircolors) are recognized but ignored.
# Below, there should be one TERM entry for each termtype that is colorizable
TERM Eterm
TERM ansi
TERM color-xterm
TERM con132x25
TERM con132x30
TERM con132x43
TERM con132x60
TERM con80x25
TERM con80x28
TERM xterm-debian
```

```
# Below are the color init strings for the basic file types. A color init
# string consists of one or more of the following numeric codes:
# Attribute codes:
# 00=none 01=bold 04=underscore 05=blink 07=reverse 08=concealed
# Text color codes:
# 30=black 31=red 32=green 33=yellow 34=blue 35=magenta 36=cyan 37=white
# Background color codes:
# 40=black 41=red 42=green 43=yellow 44=blue 45=magenta 46=cyan 47=white
#NORMAL 00 # no color code at all
#FILE 00 # regular file: use no color at all
RESET 0 # reset to “normal” color
DIR 01;34 # directory
LINK 01;36 # symbolic link. (If you set this to 'target' instead of a
# numerical value, the color is as for the file pointed to.)
HARDLINK 44;37 # regular file with more than one link
FIFO 40;33 # pipe
SOCK 01;35 # socket
DOOR 01;35 # door
BLK 40;33;01 # block device driver
CHR 40;33;01 # character device driver
ORPHAN 40;31;01 # symlink to nonexistent file, or non-stat'able file
SETUID 37;41 # file that is setuid (u+s)
SETGID 30;43 # file that is setgid (g+s)
CAPABILITY 30;41 # file with capability
STICKY_OTHER_WRITABLE 30;42 # dir that is sticky and other-writable (+t,o+w)
OTHER_WRITABLE 34;42 # dir that is other-writable (o+w) and not sticky
STICKY 37;44 # dir with the sticky bit set (+t) and not other-writable
# This is for files with execute permission:
EXEC 01;32
# List any file extensions like '.gz' or '.tar' that you would like ls
# to colorize below. Put the extension, a space, and the color init string.
# (and any comments you want to add after a '#')
# If you use DOS-style suffixes, you may want to uncomment the following:
#.cmd 01;32 # executables (bright green)
#.exe 01;32
#.com 01;32
#.btm 01;32
#.bat 01;32
# Or if you want to colorize scripts even if they do not have the
# executable bit actually set.
#.sh 01;32
#.csh 01;32
# archives or compressed (bright red)
.tar 01;31

.pcx 01;35
.mov 01;35
.mpg 01;35
.mpeg 01;35
.m2v 01;35
.mkv 01;35
.ogm 01;35
.mp4 01;35
.m4v 01;35
.mp4v 01;35
.vob 01;35
.qt 01;35
.nuv 01;35
.wmv 01;35
.asf 01;35
.rm 01;35
.rmvb 01;35
.flc 01;35
.avi 01;35
.fli 01;35
.flv 01;35
.gl 01;35
.dl 01;35
.xcf 01;35
.xwd 01;35
.yuv 01;35
# http://wiki.xiph.org/index.php/MIME_Types_and_File_Extensions
```

```
.axv 01;35
.anx 01;35
.ogv 01;35
.ogx 01;35
# audio formats
.aac 00;36
.au 00;36
.flac 00;36
.mid 00;36
.midi 00;36
.mka 00;36
.mp3 00;36
.mpc 00;36
.ogg 00;36
.ra 00;36
.wav 00;36
# http://wiki.xiph.org/index.php/MIME_Types_and_File_Extensions
.axa 00;36
.oga 00;36
.spx 00;36
.xspf 00;36
```

Linux alias命令

Linux alias命令用于设置指令的别名。

用户可利用alias，自定指令的别名。若仅输入alias，则可列出目前所有的别名设置。alias的效力仅及于该次登入的操作。若要每次登入是即自动设好别名，可在.profile或.cshrc中设定指令的别名。

语法

```
alias[别名]=[指令名称]
```

参数说明：若不加任何参数，则列出目前所有的别名设置。

实例

给命令设置别名

```
# alias lx=ls
# lx
anaconda-ks.cfg Desktop install.log install.log.syslog qte
```

Linux clear命令

Linux clear命令用于清除屏幕。

语法

```
clear
```

实例

清屏

```
#clear
```

Linux reset命令

Linux reset命令其实和 tset 是一同个命令，它的用途是设定终端机的状态。一般而言，这个命令会自动的从环境变数、命令列或是其它的组态档决定目前终端机的型态。如果指定型态是 '?' 的话，这个程序会要求使用者输入终端机的型别。

由于这个程序会将终端机设回原始的状态，除了在 login 时使用外，当系统终端机因为程序不正常执行而进入一些奇怪的状态时，你也可以用它来重设终端机o 例如不小心把二进位档用 cat 指令进到终端机，常会有终端机不再回应键盘输入，或是回应一些奇怪字元的问题。此时就可以用 reset 将终端机回复至原始状态。

语法

```
tset [-IQqrs] [-] [-e ch] [-i ch] [-k ch] [-m mapping] [terminal]
```

参数说明：

- -p　将终端机类别显示在屏幕上，但不做设定的动作。这个命令可以用来取得目前终端机的类别。
- -e ch　将 erase 字元设成 ch
- -i ch　将中断字元设成 ch
- -k ch　将删除一行的字元设成 ch
- -I　不要做设定的动作，如果没有使用选项 -Q 的话，erase、中断及删除字元的目前值依然会送到屏幕上。
- -Q　不要显示 erase、中断及删除字元的值到屏幕上。
- -r　将终端机类别印在屏幕上。
- -s　将设定 TERM 用的命令用字串的型式送到终端机中，通常在 .login 或 .profile 中用。

实例

让使用者输入一个终端机型别并将终端机设到该型别的预设状态

```
# reset ?
```

将 erase 字元设定 control-h

```
# reset -e ^B
```

将设定用的字串显示在屏幕上

```
# reset -s
Erase is control-B (^B).
Kill is control-U (^U).
Interrupt is control-C (^C).
TERM=xterm;
```

Linux enable命令

Linux enable命令用于启动或关闭 shell 内建指令。

若要执行的文件名称与shell内建指令相同，可用enable -n来关闭shell内建指令。若不加-n参数，enable可重新启动关闭的指令。

语法

```
enable [-n][-all][内建指令]
```

参数说明：

- -n 关闭指定的shell内建指令。
- -all 显示shell所有关闭与启动的指令。

实例

显示shell内置命令

```
# enable //显示shell命令
enable .
enable :
enable [
enable alias
enable bg
enable bind
enable break
enable builtin
enable caller
enable cd
enable command
enable compgen
enable complete
enable compopt
enable continue
enable declare
enable dirs
enable disown
enable echo
enable enable
enable eval
enable exec
enable exit
enable export
enable false
enable fc
enable fg
enable getopts
enable hash
enable help
enable history
enable jobs
enable kill
enable let
enable local
enable logout
enable mapfile
enable popd
enable printf
enable pushd
enable pwd
enable read
enable readarray
enable readonly
enable return
enable set
enable shift
enable shopt
enable source
enable suspend
enable test
enable times
enable trap
enable true
enable type
enable typeset
enable ulimit
enable umask
enable unalias
enable unset
enable wait
```

Linux dmesg命令

Linux dmesg命令用于显示开机信息。

kernel会将开机信息存储在ring buffer中。您若是开机时来不及查看信息，可利用dmesg来查看。开机信息亦保存在/var/log目录中，名称为dmesg的文件里。

语法

```
dmesg [-cn][-s <缓冲区大小>]
```

参数说明：

- -c 显示信息后，清除ring buffer中的内容。
- -s<缓冲区大小> 预设置为8196，刚好等于ring buffer的大小。
- -n 设置记录信息的层级。

实例

显示开机信息

```
# dmesg |less
WARNING: terminal is not fully functional
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Linux version 2.6.32-21-generic (buildd@rothera) (gcc version 4.4.3 (Ub
untu 4.4.3-4ubuntu5) ) #32-Ubuntu SMP Fri Apr 16 08:10:02 UTC 2010 (Ubuntu 2.6.32-21.3
2-generic 2.6.32.11+drm33.2)
[    0.000000] KERNEL supported cpus:
[    0.000000]  Intel GenuineIntel
[    0.000000]  AMD AuthenticAMD
[    0.000000]  NSC Geode by NSC
[    0.000000]  Cyrix CyrixInstead
[    0.000000]  Centaur CentaurHauls
[    0.000000]  Transmeta GenuineTMx86
[    0.000000]  Transmeta TransmetaCPU
[    0.000000]  UMC UMC UMC UMC
[    0.000000] BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
[    0.000000] BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
[    0.000000] BIOS-e820: 00000000000ca000 - 00000000000cc000 (reserved)
[    0.000000] BIOS-e820: 00000000000dc000 - 00000000000e0000 (reserved)
[    0.000000] BIOS-e820: 00000000000e4000 - 0000000000100000 (reserved)
[    0.000000] BIOS-e820: 0000000000100000 - 000000003fef0000 (usable)
[    0.000000] BIOS-e820: 000000003fef0000 - 000000003feff000 (ACPI data)
[    0.000000] BIOS-e820: 000000003feff000 - 000000003ff00000 (ACPI NVS)

……省略部分内容
```

显示开机信息

```
#pwd   //查看当前所在目录
/home/hnlinux/
# dmesg > boot.msg //将开机信息保存到 boot.msg文件中
#ls //显示当前目录文件
boot.msg
```

Linux depmod命令

Linux depmod命令用于分析可载入模块的相依性。

depmod(depend module)可检测模块的相依性，供modprobe在安装模块时使用。

语法

```
depmod [-adeisvV][-m <文件>][--help][模块名称]
```

参数说明：

- -a或--all　分析所有可用的模块。
- -d或debug　执行排错模式。
- -e　输出无法参照的符号。
- -i　不检查符号表的版本。
- -m<文件>或system-map<文件>　使用指定的符号表文件。
- -s或--system-log　在系统记录中记录错误。
- -v或--verbose　执行时显示详细的信息。
- -V或--version　显示版本信息。
- --help　显示帮助。

实例

显示可用模块

```
# depmod -a //显示可用模块
```

Linux declare命令

Linux declare命令用于声明 shell 变量。

declare为shell指令，在第一种语法中可用来声明变量并设置变量的属性([rix]即为变量的属性），在第二种语法中可用来显示shell函数。若不加上任何参数，则会显示全部的shell变量与函数(与执行set指令的效果相同)。

语法

```
declare [+/-][rxi][变量名称＝设置值] 或 declare -f
```

参数说明：

- +/-　"-"可用来指定变量的属性，"+"则是取消变量所设的属性。
- -f　仅显示函数。
- r　将变量设置为只读。
- x　指定的变量会成为环境变量，可供shell以外的程序来使用。
- i　[设置值]可以是数值，字符串或运算式。

实例

声明整数型变量

```
# declare -i ab //声明整数型变量
# ab=56 //改变变量内容
# echo $ab //显示变量内容
56
```

改变变量属性

```
# declare -i ef //声明整数型变量
# ef=1  //变量赋值（整数值）
# echo $ef //显示变量内容
1
# ef="wer" //变量赋值（文本值）
# echo $ef
0
# declare +i ef //取消变量属性
# ef="wer"
# echo $ef
wer
```

设置变量只读

```
# declare -r ab //设置变量为只读
# ab=88 //改变变量内容
-bash: ab: 只读变量
# echo $ab //显示变量内容
56
```

声明数组变量

```
# declare -a cd='([0]="a" [1]="b" [2]="c")' //声明数组变量
# echo ${cd[1]}
b //显示变量内容

# echo ${cd[@]} //显示整个数组变量内容
a b c
```

显示函数

```
# declare -f
command_not_found_handle ()
{
  if [ -x /usr/lib/command-not-found ]; then
    /usr/bin/python /usr/lib/command-not-found -- $1;
    return $?;
  else
    if [ -x /usr/share/command-not-found ]; then
      /usr/bin/python /usr/share/command-not-found -- $1;
      return $?;
    else
      return 127;
    fi;
  fi
}
```

Linux crontab命令

Linux crontab是用来定期执行程序的命令。

当安装完成操作系统之后，默认便会启动此任务调度命令。

crond命令每分鍾会定期检查是否有要执行的工作，如果有要执行的工作便会自动执行该工作。

而linux任务调度的工作主要分为以下两类：

- 1、系统执行的工作：系统周期性所要执行的工作，如备份系统数据、清理缓存
- 2、个人执行的工作：某个用户定期要做的工作，例如每隔10分钟检查邮件服务器是否有新信，这些工作可由每个用户自行设置

语法

```
crontab [ -u user ] file
```

或

```
crontab [ -u user ] { -l | -r | -e }
```

说明：

crontab 是用来让使用者在固定时间或固定间隔执行程序之用，换句话说，也就是类似使用者的时程表。

-u user 是指设定指定 user 的时程表，这个前提是你必须要有其权限(比如说是 root)才能够指定他人的时程表。如果不使用 -u user 的话，就是表示设定自己的时程表。

参数说明：

- -e : 执行文字编辑器来设定时程表，内定的文字编辑器是 VI，如果你想用别的文字编辑器，则请先设定 VISUAL 环境变数来指定使用那个文字编辑器(比如说 setenv VISUAL joe)
- -r : 删除目前的时程表
- -l : 列出目前的时程表

时程表的格式如下：

```
f1 f2 f3 f4 f5 program
```

- 其中 f1 是表示分钟，f2 表示小时，f3 表示一个月份中的第几日，f4 表示月份，f5 表示一个星期中的第几天。program 表示要执行的程序。
- 当 f1 为 时表示每分钟都要执行 *program，f2* 为 时表示每小时都要执行程序，其馀类推
- 当 f1 为 a-b 时表示从第 a 分钟到第 b 分钟这段时间内要执行，f2 为 a-b 时表示从第 a 到第 b 小时都要执行，其馀类推
- 当 f1 为 */n* 时表示每 *n* 分钟个时间间隔执行一次，*f2* 为 /n 表示每 n 小时个时间间隔执行一次，其馀类推
- 当 f1 为 a, b, c,... 时表示第 a, b, c,... 分钟要执行，f2 为 a, b, c,... 时表示第 a, b, c...个小时要执行，其馀类推

使用者也可以将所有的设定先存放在文件中，用 crontab file 的方式来设定时程表。

实例

每月每天每小时的第 0 分钟执行一次 /bin/ls

```
0 7 * * * /bin/ls
```

在 12 月内, 每天的早上 6 点到 12 点中，每隔 20 分钟执行一次 /usr/bin/backup

```
0 6-12/3 * 12 * /usr/bin/backup
```

周一到周五每天下午 5:00 寄一封信给 alex@domain.name

```
0 17 * * 1-5 mail -s "hi" alex@domain.name < /tmp/maildata
```

每月每天的午夜 0 点 20 分, 2 点 20 分, 4 点 20 分....执行 echo "haha"

```
20 0-23/2 * * * echo "haha"
```

下面再看看几个具体的例子：

```
0 */2 * * * /sbin/service httpd restart  意思是每两个小时重启一次apache

50 7 * * * /sbin/service sshd start  意思是每天7：50开启ssh服务

50 22 * * * /sbin/service sshd stop  意思是每天22：50关闭ssh服务

0 0 1,15 * * fsck /home  每月1号和15号检查/home 磁盘

1 * * * * /home/bruce/backup  每小时的第一分执行 /home/bruce/backup这个文件

00 03 * * 1-5 find /home "*.xxx" -mtime +4 -exec rm {} \;  每周一至周五3点钟，在目录/home中，

30 6 */10 * * ls  意思是每月的1、11、21、31日是的6：30执行一次ls命令
```

注意：当程序在你所指定的时间执行后，系统会寄一封信给你，显示该程序执行的内容，若是你不希望收到这样的信，请在每一行空一格之后加上 > /dev/null 2>&1 即可

Linux clock命令

Linux clock命令用于调整 RTC 时间。

RTC 是电脑内建的硬件时间，执行这项指令可以显示现在时刻，调整硬件时钟的时间，将系统时间设成与硬件时钟之时间一致，或是把系统时间回存到硬件时钟。

语法

```
clock [--adjust][--debug][--directisa][--getepoch][--hctosys][--set --date="<日期时间>"][-
```

参数说明：

- --adjust　第一次使用"--set"或"--systohc"参数设置硬件时钟，会在/etc目录下产生一个名称为adjtime的文件。当再次使用这两个参数调整硬件时钟，此文件便会记录两次调整间之差异，日后执行clock指令加上"--adjust"参数时，程序会自动根 据记录文件的数值差异，计算出平均值，自动调整硬件时钟的时间。
- --debug　详细显示指令执行过程，便于排错或了解程序执行的情形。
- --directisa　告诉clock指令不要通过/dev/rtc设备文件，直接对硬件时钟进行存取。这个参数适用于仅有ISA总线结构的老式电脑。
- --getepoch　把系统核心内的硬件时钟新时代数值，呈现到标准输出设备。
- --hctosys　Hardware Clock to System Time，把系统时间设成和硬件时钟一致。由于这个动作将会造成系统全面更新文件的存取时间，所以最好在系统启动时就执行它。
- --set--date　设置硬件时钟的日期和时间。
- --setepoch--epoch=<年份>　设置系统核心之硬件时钟的新时代数值，年份以四位树字表示。
- --show　读取硬件时钟的时间，并将其呈现至标准输出设备。
- --systohc　System Time to Hardware Clock，将系统时间存回硬件时钟内。
- --test　仅作测试，并不真的将时间写入硬件时钟或系统时间。
- --utc　把硬件时钟上的时间时为CUT，有时也称为UTC或UCT。
- --version　显示版本信息。

实例

获取当前的时间

```
# clock //获取当前的时间
```

显示UTC时间

```
# clock -utc //显示UTC时间
```

Linux chroot命令

Linux chroot命令用于改变根目录。

chroot(change root)命令把根目录换成指定的目的目录。

、

语法

```
chroot [--help][--version][目的目录][执行指令...]
```

参数说明：

- --help 　在线帮助。
- --version 　显示版本信息。

实例

改变根目录

```
# chroot /mnt/ls //改变根目录
```

Linux insmod命令

Linux insmod(install module)命令用于载入模块。

Linux有许多功能是通过模块的方式，在需要时才载入kernel。如此可使kernel较为精简，进而提高效率，以及保有较大的弹性。这类可载入的模块，通常是设备驱动程序。

语法

```
insmod [-fkmpsvxX][-o <模块名称>][模块文件][符号名称 = 符号值]
```

参数说明：

- -f 　不检查目前kernel版本与模块编译时的kernel版本是否一致，强制将模块载入。
- -k 　将模块设置为自动卸除。
- -m 　输出模块的载入信息。
- -o<模块名称> 　指定模块的名称，可使用模块文件的文件名。
- -p 　测试模块是否能正确地载入kernel。
- -s 　将所有信息记录在系统记录文件中。
- -v 　执行时显示详细的信息。
- -x 　不要汇出模块的外部符号。
- -X 　汇出模块所有的外部符号，此为预设置。

实例

加载模块

```
# insmod led.o
//向内核加载模块
```

Linux rpm命令

Linux rpm命令用于管理套件。

rmp(redhat package manager)原本是Red Hat Linux发行版专门用来管理Linux各项套件的程序，由于它遵循GPL规则且功能强大方便，因而广受欢迎。逐渐受到其他发行版的采用。RPM套件管理方式的出现，让Linux易于安装，升级，间接提升了Linux的适用度。

语法

```
rpm [-acdhilqRsv][-b<完成阶段><套间档>+][-e<套件挡>][-f<文件>+][-i<套件档>][-p<套件档>＋][-U<套
```

参数说明：

- -a 查询所有套件。
- -b<完成阶段><套件档>+或-t <完成阶段><套件档>+ 设置包装套件的完成阶段，并指定套件档的文件名称。
- -c 只列出组态配置文件，本参数需配合"-l"参数使用。
- -d 只列出文本文件，本参数需配合"-l"参数使用。
- -e<套件档>或--erase<套件档> 删除指定的套件。
- -f<文件>+ 查询拥有指定文件的套件。
- -h或--hash 套件安装时列出标记。
- -i 显示套件的相关信息。
- -i<套件档>或--install<套件档> 安装指定的套件档。
- -l 显示套件的文件列表。
- -p<套件档>+ 查询指定的RPM套件档。
- -q 使用询问模式，当遇到任何问题时，rpm指令会先询问用户。
- -R 显示套件的关联性信息。
- -s 显示文件状态，本参数需配合"-l"参数使用。
- -U<套件档>或--upgrade<套件档> 升级指定的套件档。
- -v 显示指令执行过程。
- -vv 详细显示指令执行过程，便于排错。
- -addsign<套件档>+ 在指定的套件里加上新的签名认证。
- --allfiles 安装所有文件。
- --allmatches 删除符合指定的套件所包含的文件。
- --badreloc 发生错误时，重新配置文件。
- --buildroot<根目录> 设置产生套件时，欲当作根目录的目录。
- --changelog 显示套件的更改记录。

- --checksig<套件档>+　检验该套件的签名认证。
- --clean　完成套件的包装后，删除包装过程中所建立的目录。
- --dbpath<数据库目录>　设置欲存放RPM数据库的目录。
- --dump　显示每个文件的验证信息。本参数需配合"-l"参数使用。
- --excludedocs　安装套件时，不要安装文件。
- --excludepath<排除目录>　忽略在指定目录里的所有文件。
- --force　强行置换套件或文件。
- --ftpproxy<主机名称或IP地址>　指定FTP代理服务器。
- --ftpport<通信端口>　设置FTP服务器或代理服务器使用的通信端口。
- --help　在线帮助。
- --httpproxy<主机名称或IP地址>　指定HTTP代理服务器。
- --httpport<通信端口>　设置HTTP服务器或代理服务器使用的通信端口。
- --ignorearch　不验证套件档的结构正确性。
- --ignoreos　不验证套件档的结构正确性。
- --ignoresize　安装前不检查磁盘空间是否足够。
- --includedocs　安装套件时，一并安装文件。
- --initdb　确认有正确的数据库可以使用。
- --justdb　更新数据库，当不变动任何文件。
- --nobulid　不执行任何完成阶段。
- --nodeps　不验证套件档的相互关联性。
- --nofiles　不验证文件的属性。
- --nogpg　略过所有GPG的签名认证。
- --nomd5　不使用MD5编码演算确认文件的大小与正确性。
- --nopgp　略过所有PGP的签名认证。
- --noorder　不重新编排套件的安装顺序，以便满足其彼此间的关联性。
- --noscripts　不执行任何安装Script文件。
- --notriggers　不执行该套件包装内的任何Script文件。
- --oldpackage　升级成旧版本的套件。
- --percent　安装套件时显示完成度百分比。
- --pipe<执行指令>　建立管道，把输出结果转为该执行指令的输入数据。
- --prefix<目的目录>　若重新配置文件，就把文件放到指定的目录下。
- --provides　查询该套件所提供的兼容度。
- --queryformat<档头格式>　设置档头的表示方式。
- --querytags　列出可用于档头格式的标签。
- --rcfile<配置文件>　使用指定的配置文件。
- --rebulid<套件档>　安装原始代码套件，重新产生二进制文件的套件。
- --rebuliddb　以现有的数据库为主，重建一份数据库。
- --recompile<套件档>　此参数的效果和指定"--rebulid"参数类似，当不产生套件档。
- --relocate<原目录>=<新目录>　把本来会放到原目录下的文件改放到新目录。
- --replacefiles　强行置换文件。

- --replacepkgs 强行置换套件。
- --requires 查询该套件所需要的兼容度。
- --resing<套件档>+ 删除现有认证，重新产生签名认证。
- --rmsource 完成套件的包装后，删除原始代码。
- --rmsource<文件> 删除原始代码和指定的文件。
- --root<根目录> 设置欲当作根目录的目录。
- --scripts 列出安装套件的Script的变量。
- --setperms 设置文件的权限。
- --setugids 设置文件的拥有者和所属群组。
- --short-circuit 直接略过指定完成阶段的步骤。
- --sign 产生PGP或GPG的签名认证。
- --target=<安装平台>+ 设置产生的套件的安装平台。
- --test 仅作测试，并不真的安装套件。
- --timecheck<检查秒数> 设置检查时间的计时秒数。
- --triggeredby<套件档> 查询该套件的包装者。
- --triggers 展示套件档内的包装Script。
- --verify 此参数的效果和指定"-q"参数相同。
- --version 显示版本信息。
- --whatprovides<功能特性> 查询该套件对指定的功能特性所提供的兼容度。
- --whatrequires<功能特性> 查询该套件对指定的功能特性所需要的兼容度。

实例

安装软件

```
# rpm -hvi dejagnu-1.4.2-10.noarch.rpm
警告：dejagnu-1.4.2-10.noarch.rpm: V3 DSA 签名：NOKEY, key ID db42a60e
准备...
########################################### [100%]
```

显示软件安装信息

```
# rpm -qi dejagnu-1.4.2-10.noarch.rpm

【第1次更新 教程、类似命令关联】
```

Linux grpconv命令

Linux grpconv(group convert to shadow password)命令用于开启群组的投影密码。

Linux系统里的用户和群组密码，分别存放在/etc目录下的passwd和group文件中。因系统运作所需，任何人都得以读取它们，造成安全上的破绽。投影密码将文件内的密码改存在/etc目录下的shadow和gshadow文件内，只允许系统管理者读取，同时把原密码置换为"x"字符。投影密码的功能可随时开启或关闭，您只需执行grpconv指令就能开启群组投影密码。

语法

```
grpconv
```

Linux pwunconv命令

Linux pwunconv命令用于关闭用户的投影密码。

执行pwunconv指令可以关闭用户投影密码，它会把密码从shadow文件内，重回存到passwd文件里。

语法

```
pwunconv
```

实例

关闭用户的投影密码

```
# pwunconv
```

Linux export命令

Linux export命令用于设置或显示环境变量。

在shell中执行程序时，shell会提供一组环境变量。export可新增，修改或删除环境变量，供后续执行的程序使用。export的效力仅及于该次登陆操作。

语法

```
export [-fnp][变量名称]=[变量设置值]
```

参数说明：

- -f 代表[变量名称]中为函数名称。
- -n 删除指定的变量。变量实际上并未删除，只是不会输出到后续指令的执行环境中。
- -p 列出所有的shell赋予程序的环境变量。

实例

列出当前所有的环境变量

```
# export -p //列出当前的环境变量值
declare -x HOME="/root"
declare -x LANG="zh_CN.UTF-8"
declare -x LANGUAGE="zh_CN:zh"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="root"
declare -x LS_COLORS=""
declare -x MAIL="/var/mail/root"
declare -x OLDPWD
declare -x PATH="/opt/toolchains/arm920t-eabi/bin:/opt/toolchains/arm920t-eabi/bin:/usr/l
declare -x PWD="/root"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SPEECHD_PORT="6560"
declare -x SSH_CLIENT="192.168.1.65 1674 22"
declare -x SSH_CONNECTION="192.168.1.65 1674 192.168.1.3 22"
declare -x SSH_TTY="/dev/pts/2"
declare -x TERM="XTERM"
declare -x USER="root"
declare -x XDG_SESSION_COOKIE="93b5d3d03e032c0cf892a4474bebda9f-1273864738.954257-3402064
```

定义环境变量

```
# export MYENV //定义环境变量
# export -p //列出当前的环境变量
declare -x HOME="/root"
declare -x LANG="zh_CN.UTF-8"
declare -x LANGUAGE="zh_CN:zh"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="root"
declare -x LS_COLORS=""
declare -x MAIL="/var/mail/root"
declare -x MYENV
declare -x OLDPWD
declare -x PATH="/opt/toolchains/arm920t-eabi/bin:/opt/toolchains/arm920t-eabi/bin:/usr/l
declare -x PWD="/root"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SPEECHD_PORT="6560"
declare -x SSH_CLIENT="192.168.1.65 1674 22"
declare -x SSH_CONNECTION="192.168.1.65 1674 192.168.1.3 22"
declare -x SSH_TTY="/dev/pts/2"
declare -x TERM="XTERM"
declare -x USER="root"
declare -x XDG_SESSION_COOKIE="93b5d3d03e032c0cf892a4474bebda9f-1273864738.954257-3402064
```

定义环境变量赋值

```
# export MYENV=7 //定义环境变量并赋值
# export -p
declare -x HOME="/root"
declare -x LANG="zh_CN.UTF-8"
declare -x LANGUAGE="zh_CN:zh"
declare -x LESSCLOSE="/usr/bin/lesspipe %s %s"
declare -x LESSOPEN="| /usr/bin/lesspipe %s"
declare -x LOGNAME="root"
declare -x LS_COLORS=""
declare -x MAIL="/var/mail/root"
declare -x MYENV="7"
declare -x OLDPWD
declare -x PATH="/opt/toolchains/arm920t-eabi/bin:/opt/toolchains/arm920t-eabi/bin:/usr/l
declare -x PWD="/root"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x SPEECHD_PORT="6560"
declare -x SSH_CLIENT="192.168.1.65 1674 22"
declare -x SSH_CONNECTION="192.168.1.65 1674 192.168.1.3 22"
declare -x SSH_TTY="/dev/pts/2"
declare -x TERM="XTERM"
declare -x USER="root"
declare -x XDG_SESSION_COOKIE="93b5d3d03e032c0cf892a4474bebda9f-1273864738.954257-3402064
```

Linux eval命令

Linux eval命令用于重新运算求出参数的内容。

eval可读取一连串的参数，然后再依参数本身的特性来执行。

语法

```
eval [参数]
```

参数说明：参数不限数目，彼此之间用分号分开。

实例

连接多个命令

```
# eval enable;ls //连接多个命令
enable .
enable :
enable [
enable alias
enable bg
enable bind
enable break
enable builtin
enable caller
enable cd
enable command
enable compgen
enable complete
enable compopt
enable continue
enable declare
enable dirs
enable disown
enable echo
enable enable
enable eval
enable exec
enable exit
enable export
enable false
enable fc
enable fg
enable getopts
enable hash
enable help
enable history
enable jobs
enable kill
enable let
enable local
enable logout
enable mapfile
enable popd
enable printf
enable pushd
enable pwd
enable read
enable readarray
enable readonly
enable return
enable set
enable shift
enable shopt
enable source
enable suspend
enable test
enable times
enable trap
enable true
enable type
enable typeset
enable ulimit
enable umask
enable unalias
enable unset
enable wait
```

Linux set命令

Linux set命令用于设置shell。

set指令能设置所使用shell的执行方式，可依照不同的需求来做设置。

语法

```
set [+-abCdefhHklmnpPtuvx]
```

参数说明：

- -a 　标示已修改的变量，以供输出至环境变量。
- -b 　使被中止的后台程序立刻回报执行状态。
- -C 　转向所产生的文件无法覆盖已存在的文件。
- -d 　Shell预设会用杂凑表记忆使用过的指令，以加速指令的执行。使用-d参数可取消。
- -e 　若指令传回值不等于0，则立即退出shell。
- -f　 　取消使用通配符。
- -h 　自动记录函数的所在位置。
- -H Shell 　可利用"!"加<指令编号>的方式来执行history中记录的指令。
- -k 　指令所给的参数都会被视为此指令的环境变量。
- -l 　记录for循环的变量名称。
- -m 　使用监视模式。
- -n 　只读取指令，而不实际执行。
- -p 　启动优先顺序模式。
- -P 　启动-P参数后，执行指令时，会以实际的文件或目录来取代符号连接。
- -t 　执行完随后的指令，即退出shell。
- -u 　当执行时使用到未定义过的变量，则显示错误信息。
- -v 　显示shell所读取的输入值。
- -x 　执行指令后，会先显示该指令及所下的参数。
- +<参数> 　取消某个set曾启动的参数。

实例

显示环境变量

```
# set
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="3" [1]="00" [2]="15" [3]="1" [4]="release" [5]="i386-redhat-linux-gnu
BASH_VERSION='3.00.15(1)-release'
COLORS=/etc/DIR_COLORS.xterm
COLUMNS=99
DIRSTACK=()
EUID=0
GROUPS=()
G_BROKEN_FILENAMES=1
HISTFILE=/root/.bash_history
HISTFILESIZE=1000
HISTSIZE=1000
HOME=/root
HOSTNAME=hnlinux
HOSTTYPE=i386
IFS=/plinux> '
INPUTRC=/etc/inputrc
KDEDIR=/usr
LANG=zh_CN.GB2312
LESSOPEN='|/usr/bin/lesspipe.sh %s'
LINES=34
L
MAIL=/var/spool/mail/root
MAILCHECK=60
OLDPWD=/home/uptech
OPTERR=1
OPTIND=1
OSTYPE=linux-gnu
PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/
PIPESTATUS=([0]="2")
PPID=26005
PROMPT_COMMAND='echo -ne "
```

Linux minfo命令

Linux minfo命令用于显示MS-DOS文件系统的各项参数。

minfo为mtools工具指令，可显示MS-DOS系统磁盘的各项参数，包括磁区数，磁头数...等。

语法

```
</>minfo [-v][驱动器代号]
```

参数说明：

- -v　除了一般信息外，并显示可开机磁区的内容。

实例

显示DOS系统参数

```
# minfo -v C: //显示系统参数
```

Linux lsmod命令

Linux lsmod命令用于显示已载入系统的模块。

执行lsmod(list modules)指令，会列出所有已载入系统的模块。Linux操作系统的核心具有模块化的特性，应此在编译核心时，务须把全部的功能都放入核心。您可以将这些功能编译成一个个单独的模块，待需要时再分别载入。

语法

```
lsmod
```

实例

显示模块信息

```
# lsmod
Module         Size Used by
nfsd          238935 11
lockd          64849 1 nfsd
nfs_acl          2245 1 nfsd
auth_rpcgss       33735 1 nfsd
sunrpc         193181 10 nfsd,lockd,nfs_acl,auth_rpcgss
exportfs         3437 1 nfsd
xt_TCPMSS         2931 1
xt_tcpmss         1197 1
xt_tcpudp         2011 1
iptable_mangle      2771 1
ip_tables         9991 1 iptable_mangle
x_tables         14299 4 xt_TCPMSS,xt_tcpmss,xt_tcpudp,ip_tables
pppoe           8943 2
pppox           2074 1 pppoe
binfmt_misc        6587 1
snd_ens1371       18814 0
gameport         9089 1 snd_ens1371
snd_ac97_codec     100646 1 snd_ens1371
ac97_bus         1002 1 snd_ac97_codec
snd_pcm_oss       35308 0
snd_mixer_oss      13746 1 snd_pcm_oss
snd_pcm         70662 3 snd_ens1371,snd_ac97_codec,snd_pcm_oss
snd_seq_dummy       1338 0
snd_seq_oss       26726 0
snd_seq_midi       4557 0
snd_rawmidi       19056 2 snd_ens1371,snd_seq_midi
snd_seq_midi_event   6003 2 snd_seq_oss,snd_seq_midi
snd_seq         47263 6 snd_seq_dummy,snd_seq_oss,snd_seq_midi,snd_seq_midi_event
snd_timer        19098 2 snd_pcm,snd_seq
snd_seq_device      5700 5 snd_seq_dummy,snd_seq_oss,snd_seq_midi,snd_rawmidi,snd_seq
fbcon          35102 71
tileblit         2031 1 fbcon
font           7557 1 fbcon
bitblit          4707 1 fbcon
ppdev           5259 0
softcursor        1189 1 bitblit
snd           54148 10 snd_ens1371,snd_ac97_codec,snd_pcm_oss,snd_mixer_oss,snd_pcm,snd_se
psmouse         63245 0
serio_raw         3978 0
soundcore         6620 1 snd
parport_pc        25962 1
snd_page_alloc      7076 1 snd_pcm
vga16fb         11385 1
intel_agp         24177 1
vgastate         8961 1 vga16fb
i2c_piix4         8335 0
shpchp          28820 0
agpgart         31724 1 intel_agp
lp           7028 0
parport         32635 3 ppdev,parport_pc,lp
mptspi          14652 2
mptscsih         31325 1 mptspi
pcnet32         28890 0
floppy          53016 0
mii           4381 1 pcnet32
mptbase         83022 2 mptspi,mptscsih
scsi_transport_spi   21096 1 mptspi
```

Linux liloconfig命令

Linux liloconfig命令用于设置核心载入，开机管理程序。

liloconfig是Slackware发行版专门用来调整lilo设置的程序。它通过互动式操作界面，让用户能够利用键盘上的方向键等，轻易地操控lilo的安装，设置作业，而无须下达各种参数或撰写配置文件。

语法

```
liloconfig
```

实例

执行liloconfig命令

```
# liloconfig
```

Linux lilo命令

Linux lilo命令用于安装核心载入，开机管理程序。

lilo(linux loader)是个Linux系统核心载入程序，同时具备管理开机的功能。单独执行lilo指令，它会读取/etc/目录下的lilo.conf配置文件，然后根据其内容安装lilo。

语法

```
lilo [-clqtV][-b<外围设备代号>][-C<配置文件>][-d<延迟时间>][－D<识别标签>][-f<几何参数文件>][-i<开
```

参数说明：

- -b<外围设备代号>　指定安装lilo之处的外围设备代号。
- -c　使用紧致映射模式。
- -C<配置文件>　指定lilo的配置文件。
- -d<延迟时间>　设置开机延迟时间。
- -D<识别标签>　指定开机后预设启动的操作系统，或系统核心识别标签。
- -f<几何参数文件>　指定磁盘的几何参数配置文件。
- -i<开机磁区文件>　指定欲使用的开机磁区文件，预设是/boot目录里的boot.b文件。
- -I<识别标签>　显示系统核心存放之处。
- -l　产生线形磁区地址。
- -m<映射文件>　指定映射文件。
- -P<fix/ignore>　决定要修复或忽略分区表的错误。
- -q　列出映射的系统核心文件。
- -r<根目录>　设置系统启动时欲挂入成为根目录的目录。
- -R<执行指令>　设置下次启动系统时，首先执行的指令。
- -s<备份文件>　指定备份文件。
- -S<备份文件>　强制指定备份文件。
- -t　不执行指令，仅列出实际执行会进行的动作。
- -u<外围色设备代号>　删除lilo。
- -U<外围设备代号>　此参数的效果和指定"-u"参数类似，当不检查时间戳记。
- -v　显示指令执行过程。
- -V　显示版本信息。

实例

安装lilo到第一台SCSI硬盘的第三个主要分区，采用3级模式。

```
# lilo -b /dev/sda3 -v -v -v
```

指定安装lilo的配置文件和备份文件。

```
# lilo -C /etc/lilo.conf2 -s /boot/boot. Backup
```

Linux kbdconfig命令

Linux kbdconfig命令用于设置键盘类型。

kbdconfig(Red Hat Linux才有的指令)是一个用来设置键盘的程序，提供图形化的操作界面。kbdconfig实际上是修改/etc/sysconfig/keyboard的键盘配置文件。

语法

```
kbdconfig [--back][--test]
```

参数：

- --back 　执行时将预设的Cancel按钮更改为Back按钮。
- --test 　仅作测试，不会实际更改设置。

实例

键盘设置：

```
# kdbconfig //设置键盘
```

Linux modprobe命令

Linux modprobe命令用于自动处理可载入模块。

modprobe可载入指定的个别模块，或是载入一组相依的模块。modprobe会根据depmod所产生的相依关系，决定要载入哪些模块。若在载入过程中发生错误，在modprobe会卸载整组的模块。

语法

```
modprobe [-acdlrtvV][--help][模块文件][符号名称 = 符号值]
```

参数：

- -a或--all 　载入全部的模块。
- -c或--show-conf 　显示所有模块的设置信息。
- -d或--debug 　使用排错模式。
- -l或--list 　显示可用的模块。
- -r或--remove 　模块闲置不用时，即自动卸载模块。
- -t或--type 　指定模块类型。
- -v或--verbose 　执行时显示详细的信息。
- -V或--version 　显示版本信息。
- -help 　显示帮助。

实例

安装软驱模块：

```
[root@w3cschool.cc ~]# modprobe -v floppy
```

卸载软驱模块：

```
[root@w3cschool.cc ~]# modprobe -v -r floppy
```

Linux ntsysv命令

Linux ntsysv命令用于设置系统的各种服务。

这是Red Hat公司遵循GPL规则所开发的程序，它具有互动式操作界面，您可以轻易地利用方向键和空格键等，开启，关闭操作系统在每个执行等级中，所要执行的系统服务。

语法

```
ntsysv [--back][--level <等级代号>]
```

参数：

- --back 　在互动式界面里，显示Back钮，而非Cancel钮。
- --level <等级代号> 　在指定的执行等级中，决定要开启或关闭哪些系统服务。

Linux mouseconfig命令

Linux mouseconfig命令用于设置鼠标相关参数。

mouseconfig为鼠标设置程序，可自动设置相关参数，或者用户也可以利用所提供互动模式自行设置鼠标。mouseconfig是Red Hat Linux才有的命令。

语法

```
mouseconfig [--back][--emulthree][--help][--expert][--kickstart][--noprobe][--test][--dev
```

参数：

- --back　在设置画面上显示Back按钮，而取代预设的Cancel按钮。
- --device<连接端口>　指定硬件连接端口。可用的选项有ttyS0，ttyS1，ttyS2，ttyS3与orpsaux。
- --emulthree　将二钮鼠标模拟成三钮鼠标。
- --help　显示帮助以及所有支持的鼠标类型。
- --expert　程序预设可自动判断部分设置值。若要自行设置，请使用--expert参数。
- --kickstart　让程序自动检测并保存所有的鼠标设置。
- --noprobe　不要检测鼠标设备。
- --test　测试模式，不会改变任何设置。

实例

以交互模式配置鼠标：

```
# mouseconfig -text
```

Linux passwd命令

Linux passwd命令用来更改使用者的密码

语法

```
passwd [-k] [-l] [-u [-f]] [-d] [-S] [username]
```

必要参数：

- -d 删除密码
- -f 强制执行
- -k 更新只能发送在过期之后
- -l 停止账号使用
- -S 显示密码信息
- -u 启用已被停止的账户
- -x 设置密码的有效期
- -g 修改群组密码
- -i 过期后停止用户账号

选择参数：

- --help 显示帮助信息
- --version 显示版本信息

实例

修改用户密码

```
# passwd w3cschool  //设置w3cschool用户的密码
Enter new UNIX password:  //输入新密码，输入的密码无回显
Retype new UNIX password:  //确认密码
passwd: password updated successfully
#
```

显示账号密码信息

```
# passwd -S w3cschool
w3cschool P 05/13/2010 0 99999 7 -1
```

删除用户密码

```
# passwd -d lx138
passwd: password expiry information changed.
```

Linux pwconv命令

Linux pwconv命令用于开启用户的投影密码。

Linux系统里的用户和群组密码，分别存放在名称为passwd和group的文件中， 这两个文件位于/etc目录下。因系统运作所需，任何人都得以读取它们，造成安全上的破绽。投影密码将文件内的密码改存在/etc目录下的shadow和gshadow文件内，只允许系统管理者读取，同时把原密码置换为"x"字符，有效的强化了系统的安全性。

语法

```
pwconv
```

实例

开启用户的投影密码

```
# pwconv
```

Linux rdate命令

Linux rdate命令用于显示其他主机的日期与时间。

执行rdate指令，向其他主机询问系统时间并显示出来。

语法

```
rdate [-ps][主机名称或IP地址...]
```

参数：

- -p 　显示远端主机的日期与时间。
- -s 　把从远端主机收到的日期和时间，回存到本地主机的系统时间。
- -u 传输协议使用UDP协议
- -l 使用syslog显示错误信息
- -t<时间> 设置超时时间

Linux resize命令

Linux resize命令设置终端机视窗的大小。

执行resize指令可设置虚拟终端机的视窗大小。

语法

```
resize [-cu][-s <列数> <行数>]
```

参数：

- -c　就算用户环境并非C Shell，也用C Shell指令改变视窗大小。
- -s <列数> <行数>　设置终端机视窗的垂直高度和水平宽度。
- -u　就算用户环境并非Bourne Shell，也用Bourne Shell指令改变视窗大小。

实例

使用 C shell

```
[root@linux w3cschool.cc]# resize -c
set noglob;
setenv COLUMNS '99';
setenv LINES '34';
unset noglob;
```

使用 Bourne shell

```
[root@hnlinux w3cschool.cc]# resize -u
COLUMNS=99;
LINES=34;
export COLUMNS LINES;
```

设置指定大小

```
[root@hnlinux w3cschool.cc]# resize -s 80 160
```

Linux rmmod命令

Linux rmmod命令用于删除模块。

执行rmmod指令，可删除不需要的模块。Linux操作系统的核心具有模块化的特性，应此在编译核心时，务须把全部的功能都放如核心。你可以将这些功能编译成一个个单独的模块，待有需要时再分别载入它们。

语法

```
rmmod [-as][模块名称...]
```

参数：

- -a 删除所有目前不需要的模块。
- -s 把信息输出至syslog常驻服务，而非终端机界面。

实例

显示已安装的模块

```
# lsmod
Module         Size Used by
cramfs         39042 1
nfsd         238935 11
lockd         64849 1 nfsd
nfs_acl         2245 1 nfsd
auth_rpcgss      33735 1 nfsd
sunrpc        193181 10 nfsd,lockd,nfs_acl,auth_rpcgss
exportfs        3437 1 nfsd
xt_TCPMSS        2931 0
xt_tcpmss        1197 0
xt_tcpudp        2011 0
iptable_mangle     2771 0
ip_tables        9991 1 iptable_mangle
x_tables        14299 4

……省略部分结果
pppoe         8943 0
pppox         2074 1 pppoe
binfmt_misc       6587 1
snd_ens1371      18814 0
gameport        9089 1 snd_ens1371
snd_ac97_codec    100646 1 snd_ens1371
ac97_bus        1002 1 snd_ac97_codec
snd_pcm_oss      35308 0
```

卸载模块

```
# rmmod -v pppoe //卸载模块pppoe
Checking ppoe for persistent data
```

安装模块

```
# insmod -v pppoe >1.log //安装模块

~# tail -b 30 1.log //显示文件信息
```

Linux grpunconv命令

Linux grpunconv命令用于关闭群组的投影密码。

执行grpunconv指令可关闭群组投影密码，它会把密码从gshadow文件内，回存到group文件里。

语法

```
grpunconv
```

实例

未关闭的情况

```
cat /etc/gshadow | grep cdy
cdy:123456::
```

关闭影子密码

```
cat /etc/gshadow
cat: /etc/gshadow: 没有那个文件或目录
```

查看密码已经复制到 /etc/group 中了。

```
cat /etc/group | grep cdy
cdy:123456:1000:
```

Linux modinfo命令

Linux modinfo命令用于显示kernel模块的信息。

modinfo会显示kernel模块的对象文件，以显示该模块的相关信息。

语法

```
modinfo [-adhpV][模块文件]
```

参数：

- -a或--author　显示模块开发人员。
- -d或--description　显示模块的说明。
- -h或--help　显示modinfo的参数使用方法。
- -p或--parameters　显示模块所支持的参数。
- -V或--version　显示版本信息。

实例

显示sg模块的信息。

```
# modinfo sg
filename:    /lib/modules/2.6.9-42.ELsmp/kernel/drivers/scsi/sg.ko
author:     Douglas Gilbert
description:  SCSI generic (sg) driver
license:    GPL
version:    3.5.31 B0B0CB1BB59F0669A1F0D6B
parm:      def_reserved_size:size of buffer reserved for each fd
parm:      allow_dio:allow direct I/O (default: 0 (disallow))
alias:     char-major-21-*
vermagic:    2.6.9-42.ELsmp SMP 686 REGPARM 4KSTACKS gcc-3.4
depends:    scsi_mod
```

Linux time命令

Linux time命令的用途，在于量测特定指令执行时所需消耗的时间及系统资源等资讯。

例如 CPU 时间、记忆体、输入输出等等。需要特别注意的是，部分资讯在 Linux 上显示不出来。这是因为在 Linux 上部分资源的分配函式与 time 指令所预设的方式并不相同，以致于 time 指令无法取得这些资料。

语法

```
time [options] COMMAND [arguments]
```

参数：

- -o 或 --output=FILE：设定结果输出档。这个选项会将 time 的输出写入 所指定的档案中。如果档案已经存在，系统将覆写其内容。
- -a 或 --append：配合 -o 使用，会将结果写到档案的末端，而不会覆盖掉原来的内容。
- -f FORMAT 或 --format=FORMAT：以 FORMAT 字串设定显示方式。当这个选项没有被设定的时候，会用系统预设的格式。不过你可以用环境变数 time 来设定这个格式，如此一来就不必每次登入系统都要设定一次。

time 指令可以显示的资源有四大项，分别是：

- Time resources
- Memory resources
- IO resources
- Command info

详细的内容如下：

1、Time Resources

E 执行指令所花费的时间，格式是：[hour]:minute:second。请注意这个数字并不代表实际的 CPU 时间。

e 执行指令所花费的时间，单位是秒。请注意这个数字并不代表实际的 CPU 时间。

S 指令执行时在核心模式（kernel mode）所花费的时间，单位是秒。

U 指令执行时在使用者模式（user mode）所花费的时间，单位是秒。

P 执行指令时 CPU 的占用比例。其实这个数字就是核心模式加上使用者模式的 CPU 时间除以总时间。

2、Memory Resources

M 执行时所占用的实体记忆体的最大值。单位是 KB

t 执行时所占用的实体记忆体的平均值，单位是 KB

K 执行程序所占用的记忆体总量（stack+data+text）的平均大小，单位是 KB

D 执行程序的自有资料区（unshared data area）的平均大小，单位是 KB

p 执行程序的自有堆叠（unshared stack）的平均大小，单位是 KB

X 执行程序间共享内容（shared text）的平均值，单位是 KB

Z 系统记忆体页的大小，单位是 byte。对同一个系统来说这是个常数

3、IO Resources

F 此程序的主要记忆体页错误发生次数。所谓的主要记忆体页错误是指某一记忆体页已经置换到置换档（swap file)中，而且已经分配给其他程序。此时该页的内容必须从置换档里再读出来。

R 此程序的次要记忆体页错误发生次数。所谓的次要记忆体页错误是指某一记忆体页虽然已经置换到置换档中，但尚未分配给其他程序。此时该页的内容并未被破坏，不必从置换档里读出来

W 此程序被交换到置换档的次数

c 此程序被强迫中断（像是分配到的 CPU 时间耗尽）的次数

w 此程序自愿中断（像是在等待某一个 I/O 执行完毕，像是磁碟读取等等）的次数

I 此程序所输入的档案数

O 此程序所输出的档案数

r 此程序所收到的 Socket Message

s 此程序所送出的 Socket Message

k 此程序所收到的信号 (Signal)数量

4、Command Info

C 执行时的参数以及指令名称

x 指令的结束代码 (Exit Status)

-p or --portability：这个选项会自动把显示格式设定成为：

real %e user %Usys %S：这么做的目的是为了与 POSIX 规格相容。

-v or --verbose：这个选项会把所有程序中用到的资源通通列出来，不但如一般英文语句，还有说明。对不想花时间去熟习格式设定或是刚刚开始接触这个指令的人相当有用。

实例

```
1\. # time date
2\. Sun Mar 26 22:45:34 GMT-8 2006
3\.
4\. real    0m0.136s
5\. user    0m0.010s
6\. sys     0m0.070s
7\. #
```

在以上实例中，执行命令"time date"(见第1行)。

系统先执行命令"date"，第2行为命令"date"的执行结果。

第3-6行为执行命令"date"的时间统计结果，其中第4行"real"为实际时间，第5行"user"为用户CPU时间，第6行"sys"为系统CPU时间。

以上三种时间的显示格式均为MMmNN[.FFF]s。

利用下面的指令

```
time -v ps -aux
```

我们可以获得执行 ps -aux 的结果和所花费的系统资源。如下面所列的资料：

```
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.4 1096 472 ? S Apr19 0:04 init
root 2 0.0 0.0 0 0 ? SW Apr19 0:00 [kflushd]
root 3 0.0 0.0 0 0 ? SW Apr19 0:00 [kpiod]
......
root 24269 0.0 1.0 2692 996 pts/3 R 12:16 0:00 ps -aux
Command being timed: "ps -aux"
User time (seconds): 0.05
System time (seconds): 0.06
Percent of CPU this job got: 68%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.16
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 0
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 238
Minor (reclaiming a frame) page faults: 46
Voluntary context switches: 0
Involuntary context switches: 0
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
```

Linux setup命令

Linux setup命令设置公用程序，是一个启动图形设置系统的命令。

setup 命令：用来配置X，打印设置，时区设置，系统服务，网络配置，配置，防火墙配置，验证配置，鼠标配置。

语法

```
setup
```

setup是一个设置公用程序，提供图形界面的操作方式。在setup中可设置7类的选项：

- 1 .登陆认证方式
- 2 .键盘组态设置
- 3 .鼠标组态设置
- 4 .开机时所要启动的系统服务
- 5 .声卡组态设置
- 6 .时区设置
- 7 .X Windows组态设置

Linux sndconfig命令

Linux sndconfig命令用于设置声卡。

sndconfig为声卡设置程序，支持PnP设置，可自动检测并设置PnP声卡。

语法

```
sndconfig [--help][--noautoconfig][--noprobe]
```

参数：

- --help 显示帮助。
- --noautoconfig 不自动设置PnP的声卡。
- --noprobe 不自动检测PnP声卡。

Linux setenv命令

Linux setenv命令用于查询或显示环境变量。

setenv为tsch中查询或设置环境变量的指令。

语法

```
setenv [变量名称][变量值]
```

实例

显示环境变量

```
setenv
```

设置环境变量

```
# setenv USER lx138
```

Linux chkconfig命令

Linux chkconfig命令用于检查，设置系统的各种服务。

这是Red Hat公司遵循GPL规则所开发的程序，它可查询操作系统在每一个执行等级中会执行哪些系统服务，其中包括各类常驻服务。

语法

```
chkconfig [--add][--del][--list][系统服务] 或 chkconfig [--level <等级代号>][系统服务][on/off
```

参数：

- --add　增加所指定的系统服务，让chkconfig指令得以管理它，并同时在系统启动的叙述文件内增加相关数据。
- --del　删除所指定的系统服务，不再由chkconfig指令管理，并同时在系统启动的叙述文件内删除相关数据。
- --level<等级代号>　指定读系统服务要在哪一个执行等级中开启或关毕。

实例

列出chkconfig所知道的所有命令。

```
# chkconfig -list
```

开启服务。

```
# chkconfig telnet on //开启Telnet服务
# chkconfig -list //列出chkconfig所知道的所有的服务的情况
```

关闭服务

```
# chkconfig telnet off  //关闭Telnet服务
# chkconfig -list //列出chkconfig所知道的所有的服务的情况
```

Linux unset命令

Linux unset命令用于删除变量或函数。

unset为shell内建指令，可删除变量或函数。

语法

```
unset [-fv][变量或函数名称]
```

参数：

- -f 仅删除函数。
- -v 仅删除变量。

实例

删除环境变量

```
[root@w3cschool.cc ~]# lx="ls -lh" //设定环境变量
[root@w3cschool.cc ~]# $lx //使用环境变量
总用量 116K
-rw-r--r-- 1 root root 2.1K 2008-03-30 anaconda-ks.cfg
drwx------ 3 root root 4.0K 3月 30 21:22 Desktop
-rw-r--r-- 1 root root 50K 2008-03-30 install.log
-rw-r--r-- 1 root root 32K 2008-03-30 install.log.syslog
lrwxrwxrwx 1 root root  9 2008-03-30 qte -> /opt/qte/
[root@w3cschool.cc ~]# set //查看当前的环境变量
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
……省略部分内容
PROMPT_COMMAND='echo -ne "33]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}07"'
PS1='[u@h W]$ '
PS2='> '
PS4='+ '
PWD=/root
QTDIR=/usr/lib/qt-3.3
SHELL=/bin/bash
SSH_TTY=/dev/pts/4
SUPPORTED=zh_CN.UTF-8:zh_CN:zh:en_US.UTF-8:en_US:en
SYSFONT=latarcyrheb-sun16
TERM=xterm
UID=0
USER=root
_=-lh
lx='ls -lh'
[root@w3cschool.cc ~]# unset lx //删除环境变量
[root@w3cschool.cc ~]# set //显示当前环境变量
BASH=/bin/bash
BASH_ARGC=()
BASH_ARGV=()
……省略部分内容
PROMPT_COMMAND='echo -ne "33]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}07"'
PS1='[u@h W]$ '
PS2='> '
PS4='+ '
PWD=/root
QTDIR=/usr/lib/qt-3.3
SHELL=/bin/bash
SSH_TTY=/dev/pts/4
SUPPORTED=zh_CN.UTF-8:zh_CN:zh:en_US.UTF-8:en_US:en
SYSFONT=latarcyrheb-sun16
TERM=xterm
UID=0
USER=root
_=-lh
```

Linux ulimit命令

Linux ulimit命令用于控制shell程序的资源。

ulimit为shell内建指令，可用来控制shell执行程序的资源。

语法

```
ulimit [-aHS][-c <core文件上限>][-d <数据节区大小>][-f <文件大小>][-m <内存大小>][-n <文件数目>]
```

参数：

- -a 　显示目前资源限制的设定。
- -c <core文件上限> 　设定core文件的最大值，单位为区块。
- -d <数据节区大小> 　程序数据节区的最大值，单位为KB。
- -f <文件大小> 　shell所能建立的最大文件，单位为区块。
- -H 　设定资源的硬性限制，也就是管理员所设下的限制。
- -m <内存大小> 　指定可使用内存的上限，单位为KB。
- -n <文件数目> 　指定同一时间最多可开启的文件数。
- -p <缓冲区大小> 　指定管道缓冲区的大小，单位512字节。
- -s <堆叠大小> 　指定堆叠的上限，单位为KB。
- -S 　设定资源的弹性限制。
- -t <CPU时间> 　指定CPU使用时间的上限，单位为秒。
- -u <程序数目> 　用户最多可开启的程序数目。
- -v <虚拟内存大小> 　指定可使用的虚拟内存上限，单位为KB。

实例

显示系统资源的设置

```
[root@w3cschool.cc ~]# ulimit -a
core file size     (blocks, -c) 0
data seg size      (kbytes, -d) unlimited
file size        (blocks, -f) unlimited
pending signals         (-i) 1024
max locked memory    (kbytes, -l) 32
max memory size     (kbytes, -m) unlimited
open files           (-n) 1024
pipe size      (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
stack size       (kbytes, -s) 10240
cpu time        (seconds, -t) unlimited
max user processes       (-u) 4096
virtual memory     (kbytes, -v) unlimited
file locks           (-x) unlimited
[root@w3cschool.cc ~]#
```

设置单一用户程序数目上限

```
[root@w3cschool.cc ~]# ulimit -u 500 //设置单一用户程序上限
[root@w3cschool.cc ~]# ulimit -a
core file size     (blocks, -c) 0
data seg size      (kbytes, -d) unlimited
file size        (blocks, -f) unlimited
pending signals         (-i) 1024
max locked memory    (kbytes, -l) 32
max memory size     (kbytes, -m) unlimited
open files           (-n) 1024
pipe size      (512 bytes, -p) 8
POSIX message queues   (bytes, -q) 819200
stack size       (kbytes, -s) 10240
cpu time        (seconds, -t) unlimited
max user processes       (-u) 500
virtual memory     (kbytes, -v) unlimited
file locks           (-x) unlimited
[root@w3cschool.cc ~]#
```

Linux timeconfig命令

Linux timeconfig命令用于设置时区。

这是Red Hat公司遵循GPL规则所开发的程序，它具有互动式操作界面，您可以轻易地利用方向键和空格键等，设置系统时间所属的时区。

语法

```
timeconfig [--arc][--back][--test][--utc][时区名称]
```

参数：

- --arc　使用Alpha硬件结构的格式存储系统时间。
- --back　在互动式界面里，显示Back钮而非Cancel钮。
- --test　仅作测试，并不真的改变系统的时区。
- --utc　把硬件时钟上的时间视为CUT，有时也称为UTC或UCT。

实例

```
# timeconfig //设置时区
```

Linux setconsole命令

Linux setconsole命令用于设置系统终端。

setconsole可用来指定系统终端。

语法

```
setconsole [serial][ttya][ttyb]
```

参数：

- serial 使用PROM终端。
- ttya,cua0或ttyS0 使用第1个串口设备作为终端。
- ttyb,cua1或ttyS1 使用第2个串口设备作为终端。
- video 使用主机上的现卡作为终端。

实例

设置终端

```
# setconsole ttyS0
```

Linux mkkickstart命令

Linux mkkickstart命令用于建立安装的组态文件。

mkkickstart可根据目前系统的设置来建立组态文件，供其他电脑在安装时使用。组态文件的内容包括使用语言，网络环境，系统磁盘状态，以及X Windows的设置等信息。

语法

```
mkkickstart [--bootp][--dhcp][--nonet][--nox][--version][--nfs <远端电脑:路径>]
```

参数：

- --bootp 安装与开机时，使用BOOTP。
- --dhcp 安装与开机时，使用DHCP。
- --nfs<远端电脑:路径> 使用指定的网络路径安装。
- --nonet 不要进行网络设置，即假设在没有网络环境的状态下。
- --nox 不要进行X Windows的环境设置。
- --version 显示版本信息。

实例

构建一个安装组态文件：

```
# mkkickstart --nonet -bootp
```

Linux hwclock命令

Linux hwclock命令用于显示与设定硬件时钟。

在Linux中有硬件时钟与系统时钟等两种时钟。硬件时钟是指主机板上的时钟设备，也就是通常可在BIOS画面设定的时钟。系统时钟则是指kernel中的时钟。当Linux启动时，系统时钟会去读取硬件时钟的设定，之后系统时钟即独立运作。所有Linux相关指令与函数都是读取系统时钟的设定。

语法

```
hwclock [--adjust][--debug][--directisa][--hctosys][--show][--systohc][--test]
[--utc][--version][--set --date=<日期与时间>]
```

参数：

- --adjust　hwclock每次更改硬件时钟时，都会记录在/etc/adjtime文件中。使用--adjust参数，可使hwclock根据先前的记录来估算硬件时钟的偏差，并用来校正目前的硬件时钟。
- --debug　显示hwclock执行时详细的信息。
- --directisa　hwclock预设从/dev/rtc设备来存取硬件时钟。若无法存取时，可用此参数直接以I/O指令来存取硬件时钟。
- --hctosys　将系统时钟调整为与目前的硬件时钟一致。
- --set --date=<日期与时间>　设定硬件时钟。
- --show　显示硬件时钟的时间与日期。
- --systohc　将硬件时钟调整为与目前的系统时钟一致。
- --test　仅测试程序，而不会实际更改硬件时钟。
- --utc　若要使用格林威治时间，请加入此参数，hwclock会执行转换的工作。
- --version　显示版本信息。

实例

显示当前时间

```
# hwclock
2010年05月27日 星期四 18时04分31秒 -0.704214 seconds
```

查看版本信息

```
# hwclock -v
hwclock from util-linux-2.12a
```

Linux apmd命令

Linux apmd命令用于进阶电源管理服务程序。

apmd负责BIOS进阶电源管理(APM)相关的记录，警告与管理工作。

语法

```
apmd [-u v V W][-p <百分比变化量>][-w <百分比值>]
```

参数：

- -p<百分比变化量>或--percentage<百分比变化量> 当电力变化的幅度超出设置的百分比变化量，即记录事件百分比变化量的预设值为5，若设置值超过100，则关闭此功能。
- -u或--utc 将BIOS时钟设为UTC，以便从悬待模式恢复时，将-u参数传送至clock或hwclock程序。
- -v或--verbose 记录所有的APM事件。
- -V或--version 显示版本信息。
- -w<百分比值>或--warn<百分比值> 当电池不在充电状态时，且电池电量低于设置的百分比值，则在syslog(2)的ALERT层记录警告信息。百分比值的预设置为10，若设置为0，则关闭此功能。
- -W或--wall 发出警告信息给所有人。

实例

记录所有的电源管理事件

```
# apmd -v
```

设置BIOS时钟

```
# apmd -utc //设置BIOS时钟为UTC
```

Linux fbset命令

Linux fbset命令用于设置景框缓冲区。

fbset指令可用于设置景框缓冲区的大小，还能调整画面之分辨率，位置，高低宽窄，色彩 深度，并可决定是否启动先卡之各项硬件特性。

语法

```
fbset [-ahinsvVx][-db <信息文件>][-fb <外围设备代号>][--test][显示模式]
```

参数：

- -a或--all　改变所有使用该设备之虚拟终端机的显示模式。
- -db<信息文件>　指定显示模式的信息文件，预设值文件名称为fb.modes，存放在/etc目录下
- -fb<外围设备代号>　指定用来做为输出景框缓冲区之外围设备，预设置为"/dev/fd0"。
- -h或-help　在线帮助。
- -i或--info　列出所有景框缓冲区之相关信息。
- -ifb<外围设备代号>　使用另一个景框缓冲区外围设备之设置值。
- -n或--now　马上改变显示模式。
- -ofb<外围设备代号>　此参数效果和指定"-fb"参数相同。
- -s或--show　列出目前显示模式之设置。
- -v或--verbose　显示指令执行过程。
- -V或--version　显示版本信息。
- -x或--xfree86　使用XFree86兼容模式。
- --test　仅做测试，并不改变现行的显示模式。

实例

设置画面分辨率 和桌面分辨率

```
# fbset -g 800 688 1024 768//画面分辨率为800*600 桌面分辨率为1024*768
```

启动硬件文本加速

```
# fbset -accel true  // 启动硬件文本加速
```

启动广播功能

```
# fbset -bcast true //启动广播功能
```

Linux unalias命令

Linux unalias命令用于删除别名。

unalias为shell内建指令，可删除别名设置。

语法

```
unalias [-a][别名]
```

参数：

- -a   删除全部的别名。

实例

给命令设置别名

```
[root@w3cschool.cc ~]# alias lx=ls
[root@w3cschool.cc ~]# lx
anaconda-ks.cfg Desktop install.log install.log.syslog qte
```

删除别名

```
[root@w3cschool.cc ~]# alias lx //显示别名
alias lx='ls'
[root@w3cschool.cc ~]# unalias lx //删除别名
[root@w3cschool.cc ~]# lx
-bash: lx: command not found
```

Linux SVGATextMode命令

Linux SVGATextMode命令用于加强文字模式的显示画面。

SVGATextMode可用来设置文字模式下的显示画面，包括分辨率，字体和更新频率等。

语法

```
SVGATextMode [-acdfhmnrsv][-t <配置文件>][模式]
```

参数：

- -a　如果新显示模式的屏幕大小与原先不同时，SVGATextMode会执行必要的系统设置。
- -c　维持原有的VGA时脉。
- -d　执行时会显示详细的信息，供排错时参考。
- -f　不要执行配置文件中有关字体载入的指令。
- -h　显示帮助。
- -m　允许1x1的方式来重设屏幕大小。
- -n　仅测试指定的模式。
- -r　通知或重设与屏幕大小相关的程序。
- -s　显示配置文件中所有可用的模式。
- -t<配置文件>　指定配置文件。
- -v　SVGATextMode在配置新的显示模式时，预设会先检查垂直与水平的更新更新频率是否在配置文件所指定的范围内，如果不在范围内，则不设置新的显示模式。
- 模式］　[模式]参数必须是配置文件中模式的名称。

Linux命令大全 - 备份压缩

ar	bunzip2	bzip2	bzip2recover
gunzip	unarj	compress	cpio
dump	uuencode	gzexe	gzip
lha	restore	tar	uudecode
unzip	zip	zipinfo	

Linux bzip2recover命令

Linux bzip2recover命令用来修复损坏的.bz2文件。

bzip2是以区块的方式来压缩文件，每个区块视为独立的单位。因此，当某一区块损坏时，便可利用bzip2recover，试着将文件中的区块隔开来，以便解压缩正常的区块。通常只适用在压缩文件很大的情况。

语法

```
bzip2recover [.bz2 压缩文件]
```

实例

修复.bz2文件

bzip2recover col.bz2

Linux bzip2命令

Linux bzip2命令是.bz2文件的压缩程序。

bzip2采用新的压缩演算法，压缩效果比传统的LZ77/LZ78压缩演算法来得好。若没有加上任何参数，bzip2压缩完文件后会产生.bz2的压缩文件，并删除原始的文件。

语法

```
bzip2 [-cdfhkLstvVz][--repetitive-best][--repetitive-fast][- 压缩等级][要压缩的文件]
```

参数：

- -c或--stdout　将压缩与解压缩的结果送到标准输出。
- -d或--decompress　执行解压缩。
- -f或--force　bzip2在压缩或解压缩时，若输出文件与现有文件同名，预设不会覆盖现有文件。若要覆盖，请使用此参数。
- -h或--help　显示帮助。
- -k或--keep　bzip2在压缩或解压缩后，会删除原始的文件。若要保留原始文件，请使用此参数。
- -s或--small　降低程序执行时内存的使用量。
- -t或--test　测试.bz2压缩文件的完整性。
- -v或--verbose　压缩或解压缩文件时，显示详细的信息。
- -z或--compress　强制执行压缩。
- -L,--license,
- -V或--version　显示版本信息。
- --repetitive-best　若文件中有重复出现的资料时，可利用此参数提高压缩效果。
- --repetitive-fast　若文件中有重复出现的资料时，可利用此参数加快执行速度。
- -压缩等级　压缩时的区块大小。

实例

解压.bz2文件

```
[root@w3cschool.cc ~]# bzip2 -v temp.bz2 //解压文件显示详细处理信息
```

压缩文件

```
[root@w3cschool.cc ~]# bzip2 -c a.c b.c c.c
```

检查文件完整性

```
[root@w3cschool.cc ~]# bzip2 -t temp.bz2
```

Linux bunzip2命令

Linux bunzip2命令是.bz2文件的解压缩程序。

bunzip2可解压缩.bz2格式的压缩文件。bunzip2实际上是bzip2的符号连接，执行bunzip2与bzip2 -d的效果相同。

语法：bunzip2 [-fkLsvV][.bz2压缩文件]

参数：

- -f或--force 解压缩时，若输出的文件与现有文件同名时，预设不会覆盖现有的文件。若要覆盖，请使用此参数。
- -k或--keep 在解压缩后，预设会删除原来的压缩文件。若要保留压缩文件，请使用此参数。
- -s或--small 降低程序执行时，内存的使用量。
- -v或--verbose 解压缩文件时，显示详细的信息。
- -l,--license,-V或--version 显示版本信息。

实例

解压.bz2文件

```
# bunzip2 -v temp.bz2 //解压文件显示详细处理信息
```

Linux ar命令

Linux ar命令用于建立或修改备存文件，或是从备存文件中抽取文件。

ar可让您集合许多文件，成为单一的备存文件。在备存文件中，所有成员文件皆保有原来的属性与权限。

语法

```
ar[-dmpqrtx][cfosSuvV][a<成员文件>][b<成员文件>][i<成员文件>][备存文件][成员文件]
```

参数：

必要参数：

- -d 删除备存文件中的成员文件。
- -m 变更成员文件在备存文件中的次序。
- -p 显示备存文件中的成员文件内容。
- -q 将问家附加在备存文件末端。
- -r 将文件插入备存文件中。
- -t 显示备存文件中所包含的文件。
- -x 自备存文件中取出成员文件。

选项参数：

- a<成员文件> 将文件插入备存文件中指定的成员文件之后。
- b<成员文件> 将文件插入备存文件中指定的成员文件之前。
- c 建立备存文件。
- f 为避免过长的文件名不兼容于其他系统的ar指令指令，因此可利用此参数，截掉要放入备存文件中过长的成员文件名称。
- i<成员文件> 将问家插入备存文件中指定的成员文件之前。
- o 保留备存文件中文件的日期。
- s 若备存文件中包含了对象模式，可利用此参数建立备存文件的符号表。
- S 不产生符号表。
- u 只将日期较新文件插入备存文件中。
- v 程序执行时显示详细的信息。
- V 显示版本信息。

实例

打包文件

```
[root@w3cschool.cc ~]# ls   //显示当前目录文件
a.c    b.c d.c   install.log      qte
anaconda-ks.cfg c.c Desktop

[root@w3cschool.cc ~]# ar rv one.bak a.c b.c //打包 a.c b.c文件
ar: 正在创建 one.bak
a - a.c
a - b.c
[root@w3cschool.cc ~]#
```

打包多个文件

```
[root@w3cschool.cc ~]# ar rv two.bak *.c  //打包以.c结尾的文件
ar: 正在创建 two.bak
a - a.c
a - b.c
a - c.c
a - d.c
[root@w3cschool.cc ~]#
```

显示打包文件的内容

```
[root@w3cschool.cc ~]# ar t two.bak
a.c
b.c
c.c
d.c
[root@w3cschool.cc ~]#
```

删除打包文件的成员文件

```
[root@w3cschool.cc ~]# ar d two.bak a.c b.c c.c
[root@w3cschool.cc ~]# ar t two.bak
d.c
```

Linux gunzip命令

Linux gunzip命令用于解压文件。

gunzip是个使用广泛的解压缩程序，它用于解开被gzip压缩过的文件，这些压缩文件预设最后的扩展名为".gz"。事实上gunzip就是gzip的硬连接，因此不论是压缩或解压缩，都可通过gzip指令单独完成。

语法

参数：

```
gunzip [-acfhlLnNqrtvV][-s <压缩字尾字符串>][文件...] 或 gunzip [-acfhlLnNqrtvV][-s <压缩字尾
```

- -a或--ascii 使用ASCII文字模式。
- -c或--stdout或--to-stdout 把解压后的文件输出到标准输出设备。
- -f或-force 强行解开压缩文件，不理会文件名称或硬连接是否存在以及该文件是否为符号连接。
- -h或--help 在线帮助。
- -l或--list 列出压缩文件的相关信息。
- -L或--license 显示版本与版权信息。
- -n或--no-name 解压缩时，若压缩文件内含有远来的文件名称及时间戳记，则将其忽略不予处理。
- -N或--name 解压缩时，若压缩文件内含有原来的文件名称及时间戳记，则将其回存到解开的文件上。
- -q或--quiet 不显示警告信息。
- -r或--recursive 递归处理，将指定目录下的所有文件及子目录一并处理。
- -S<压缩字尾字符串>或--suffix<压缩字尾字符串> 更改压缩字尾字符串。
- -t或--test 测试压缩文件是否正确无误。
- -v或--verbose 显示指令执行过程。
- -V或--version 显示版本信息。

实例

```
<p>解压文件
</p>
<pre>
# gunzip ab.gz
```

Linux unarj命令

Linux unarj命令用于解压缩.arj文件。

unarj为.arj压缩文件的压缩程序。

语法

```
unarj [eltx][.arj压缩文件]
```

参数：

- e 解压缩.arj文件。
- l 显示压缩文件内所包含的文件。
- t 检查压缩文件是否正确。
- x 解压缩时保留原有的路径。

实例

解压.arj文件

```
# unarj e test.arj
```

Linux compress命令

Linux compress命令是一个相当古老的 unix 档案压缩指令，压缩后的档案会加上一个 .Z 延伸档名以区别未压缩的档案，压缩后的档案可以以 uncompress 解压。若要将数个档案压成一个压缩档，必须先将档案 tar 起来再压缩。由于 gzip 可以产生更理想的压缩比例，一般人多已改用 gzip 为档案压缩工具。

语法

```
compress [-dfvcV] [-b maxbits] [file ...]
```

参数：

- c 输出结果至标准输出设备（一般指荧幕）
- f 强迫写入档案，若目的档已经存在，则会被覆盖 (force)
- v 将程序执行的讯息印在荧幕上 (verbose)
- b 设定共同字串数的上限，以位元计算，可以设定的值为 9 至 16 bits 。由于值越大，能使用的共同字串就 越多，压缩比例就越大，所以一般使用预设值 16 bits (bits)
- d 将压缩档解压缩
- V 列出版本讯息
- 范例：
- 将 source.dat 压缩成 source.dat.Z ，若 source.dat.Z 已经存在，内容则会被压缩档覆盖。
- compress -f source.dat
- 将 source.dat 压缩成 source.dat.Z ，并列印出压缩比例。
- -v 与 -f 可以一起使用
- compress -vf source.dat
- 将压缩后的资料输出后再导入 target.dat.Z 可以改变压缩档名。
- compress -c source.dat > target.dat.Z
- -b 的值越大，压缩比例就越大，范围是 9-16 ，预设值是 16 。
- compress -b 12 source.dat
- 将 source.dat.Z 解压成 source.dat ，若档案已经存在，使用者按 y 以确定覆盖档案，若使用 -df 程序则会自动覆盖档案。由于系统会自动加入 .Z 为延伸档名，所以 source.dat 会自动当作 source.dat.Z 处理。
- compress -d source.dat
- compress -d source.dat.Z

压缩文件

```
[root@w3cschool.cc ~]# compress abc.h
[root@w3cschool.cc ~]# ls

abc.h.Z
```

解压文件

```
[root@w3cschool.cc ~]# compress -d abc.h.Z
[root@w3cschool.cc ~]# ls

abc.h.
```

按指定压缩比例进行压缩

```
[root@w3cschool.cc ~]# compress -b 7 abc.h
```

强制压缩文件夹

```
[root@w3cschool.cc ~]# compress -rf /home/abc/
```

Linux cpio命令

Linux cpio命令用于备份文件。

cpio是用来建立，还原备份档的工具程序，它可以加入，解开cpio或tra备份档内的文件。

语法

```
cpio [-0aABckLovV][-C <输入/输出大小>][-F <备份档>][-H <备份格式>][-O <备份档>][--block-size=<
```

参数：

- -0或--null 　接受新增列控制字符，通常配合find指令的"-print0"参数使用。
- -a或--reset-access-time 　重新设置文件的存取时间。
- -A或--append 　附加到已存在的备份档中，且这个备份档必须存放在磁盘上，而不能放置于磁带机里。
- -b或--swap 　此参数的效果和同时指定"-sS"参数相同。
- -B 　将输入/输出的区块大小改成5210 Bytes。
- -c 　使用旧ASCII备份格式。
- -C<区块大小>或--io-size=<区块大小> 　设置输入/输出的区块大小，单位是Byte。
- -d或--make-directories 　如有需要cpio会自行建立目录。
- -E<范本文件>或--pattern-file=<范本文件> 　指定范本文件，其内含有一个或多个范本样式，让cpio解开符合范本条件的文件，格式为每列一个范本样式。
- -f或--nonmatching 　让cpio解开所有不符合范本条件的文件。
- -F<备份档>或--file=<备份档> 　指定备份档的名称，用来取代标准输入或输出，也能借此通过网络使用另一台主机的保存设备存取备份档。
- -H<备份格式> 　指定备份时欲使用的文件格式。
- -i或--extract 　执行copy-in模式，还原备份档。
- -I<备份档> 　指定备份档的名称，用来取代标准输入，也能借此通过网络使用另一台主机的保存设备读取备份档。
- -k 　此参数将忽略不予处理，仅负责解决cpio不同版本间的兼容性问题。
- -l或--link 　以硬连接的方式取代复制文件，可在copy-pass模式下运用。
- -L或--dereference 　不建立符号连接，直接复制该连接所指向的原始文件。
- -m或preserve-modification-time 　不去更换文件的更改时间。
- -M<回传信息>或--message=<回传信息> 　设置更换保存媒体的信息。
- -n或--numeric-uid-gid 　使用"-tv"参数列出备份档的内容时，若再加上参数"-n"，则会以用户识别码和群组识别码替代拥有者和群组名称列出文件清单。
- -o或--create 　执行copy-out模式，建立备份档。

- -O<备份档>　指定备份档的名称，用来取代标准输出，也能借此通过网络　使用另一台主机的保存设备存放备份档。
- -p或--pass-through　执行copy-pass模式，略过备份步骤，直接将文件复制到目的目录。
- -r或--rename　当有文件名称需要更动时，采用互动模式。
- -R<拥有者><:/.><所属群组>或
- ----owner<拥有者><:/.><所属群组>　在copy-in模式还原备份档，或copy-pass模式复制文件时，可指定这些备份，复制的文件的拥有者与所属群组。
- -s或--swap-bytes　交换每对字节的内容。
- -S或--swap-halfwords　交换每半个字节的内容。
- -t或--list　将输入的内容呈现出来。
- -u或--unconditional　置换所有文件，不论日期时间的新旧与否，皆不予询问而直接覆盖。
- -v或--verbose　详细显示指令的执行过程。
- -V或--dot　执行指令时，在每个文件的执行程序前面加上"."号
- --block-size=<区块大小>　设置输入/输出的区块大小，假如设置数值为5，则区块大小为2500，若设置成10，则区块大小为5120，依次类推。
- --force-local　强制将备份档存放在本地主机。
- --help　在线帮助。
- --no-absolute-filenames　使用相对路径建立文件名称。
- --no-preserve-owner　不保留文件的拥有者，谁解开了备份档，那些文件就归谁所有。
- -only-verify-crc　当备份档采用CRC备份格式时，可使用这项参数检查备份档内的每个文件是否正确无误。
- --quiet　不显示复制了多少区块。
- --sparse　倘若一个文件内含大量的连续0字节，则将此文件存成稀疏文件。
- --version　显示版本信息。

实例

制作备份文件

```
[root@w3cschool.cc var]# ll //显示当前目录下的文件
总用量 164
drwxr-xr-x  2 root  root  4096 2008-03-30 account
drwxr-xr-x  9 root  root  4096 2008-03-30 cache
drwxr-xr-x  3 netdump netdump 4096 2008-03-30 crash
drwxr-xr-x  3 root  root  4096 2008-03-30 db
drwxr-xr-x  3 root  root  4096 2008-03-30 empty
drwxr-xr-x  3 root  root  4096 2008-03-30 ftp
drwxrwx--T  2 root  gdm   4096 4月 9 20:17 gdm
drwxr-xr-x 25 root  root  4096 2008-03-30 lib
drwxr-xr-x  2 root  root  4096 2004-08-13 local
drwxrwxr-x  6 root  lock  4096 5月 8 15:25 lock
drwxr-xr-x 14 root  root  4096 5月 8 15:14 log
lrwxrwxrwx  1 root  root   10 2008-03-30 mail -> spool/mail
drwxr-xr-x  2 root  root  4096 2004-08-13 nis
drwxr-xr-x  2 root  root  4096 2004-08-13 opt
drwxr-xr-x  2 root  root  4096 2004-08-13 preserve
drwxr-xr-x 16 root  root  4096 5月 8 15:14 run
drwxr-xr-x 16 root  root  4096 2008-03-30 spool
drwxrwxrwt  3 root  root  4096 1月 13 18:53 tmp
drwx------  2 root  root  4096 2004-07-08 tux
drwxr-xr-x  8 root  root  4096 1月 19 19:39 www
drwxr-xr-x  3 root  root  4096 2008-03-30 yp
[root@w3cschool.cc var]# ls | cpio -o >123.cpio //制作备份文件
25 blocks
[root@w3cschool.cc var]# ll //显示当前目录下的文件
总用量 172
-rw-r--r--  1 root  root  1024 5月 24 13:06 123.cpio
drwxr-xr-x  2 root  root  4096 2008-03-30 account
drwxr-xr-x  9 root  root  4096 2008-03-30 cache
drwxr-xr-x  3 netdump netdump 4096 2008-03-30 crash
drwxr-xr-x  3 root  root  4096 2008-03-30 db
drwxr-xr-x  3 root  root  4096 2008-03-30 empty
drwxr-xr-x  3 root  root  4096 2008-03-30 ftp
drwxrwx--T  2 root  gdm   4096 4月 9 20:17 gdm
drwxr-xr-x 25 root  root  4096 2008-03-30 lib
drwxr-xr-x  2 root  root  4096 2004-08-13 local
drwxrwxr-x  6 root  lock  4096 5月 8 15:25 lock
drwxr-xr-x 14 root  root  4096 5月 8 15:14 log
lrwxrwxrwx  1 root  root   10 2008-03-30 mail -> spool/mail
drwxr-xr-x  2 root  root  4096 2004-08-13 nis
drwxr-xr-x  2 root  root  4096 2004-08-13 opt
drwxr-xr-x  2 root  root  4096 2004-08-13 preserve
drwxr-xr-x 16 root  root  4096 5月 8 15:14 run
drwxr-xr-x 16 root  root  4096 2008-03-30 spool
drwxrwxrwt  3 root  root  4096 1月 13 18:53 tmp
drwx------  2 root  root  4096 2004-07-08 tux
drwxr-xr-x  8 root  root  4096 1月 19 19:39 www
drwxr-xr-x  3 root  root  4096 2008-03-30 yp
[root@w3cschool.cc var]#
```

解压备份文件

```
[root@w3cschool.cc var]# ls | cpio -i -l 123.cpio
```

解压缩备份文件，并列出详细信息

```
[root@w3cschool.cc var]# cpio -t -I 123.cpio
123.cpio
a.c
b.c
c.c
……省略部分结果
```

强制解压缩

```
[root@w3cschool.cc var]# cpio -i -u -I 123.cpio
```

解压缩时进行反向匹配，指定不解压的文件

```
[root@w3cschool.cc var]# cpio -i -I 123.cpio -f *.c

//不解压.c结尾的文件
```

向指定的.cpio文件添加文件

```
[root@w3cschool.cc var]# ls
123.cpio crash ftp local mail preserve tmp yp
account  db    gdm lock  nis  run       tux
cache     empty lib log  opt  spool    www
[root@w3cschool.cc var]# cpio -o -O 123.cpio -A
db  //用户输入 按下Ctrl+D结束输入
1 block
[root@w3cschool.cc var]#
```

从标准输入备份文件

```
[root@w3cschool.cc test]# ls
a. a.c b.c c.c d.c    f.c
[root@w3cschool.cc test]# cpio -o >123.cpio
a.c  //用户输入
b.c
c.c  //按下Ctrl+D完成输入
3 block
[root@w3cschool.cc test]#
```

复制文件

```
[root@w3cschool.cc test]# cpio -p /root
a.c  //用户输入
b.c
c.c  //按下Ctrl+D完成输入
3 block
```

Linux dump命令

Linux dump命令用于备份文件系统。

dump为备份工具程序，可将目录或整个文件系统备份至指定的设备，或备份成一个大文件。

语法

```
dump [-cnu][-0123456789][-b <区块大小>][-B <区块数目>][-d <密度>][-f <设备名称>][-h <层级>][-s
```

参数：

- -0123456789 备份的层级。
- -b<区块大小> 指定区块的大小，单位为KB。
- -B<区块数目> 指定备份卷册的区块数目。
- -c 修改备份磁带预设的密度与容量。
- -d<密度> 设置磁带的密度。单位为BPI。
- -f<设备名称> 指定备份设备。
- -h<层级> 当备份层级等于或大雨指定的层级时，将不备份用户标示为"nodump"的文件。
- -n 当备份工作需要管理员介入时，向所有"operator"群组中的使用者发出通知。
- -s<磁带长度> 备份磁带的长度，单位为英尺。
- -T<日期> 指定开始备份的时间与日期。
- -u 备份完毕后，在/etc/dumpdates中记录备份的文件系统，层级，日期与时间等。
- -w 与-W类似，但仅显示需要备份的文件。
- -W 显示需要备份的文件及其最后一次备份的层级，时间与日期。

实例

备份文件到磁带

```
# dump -0 -u /dev/tape /home/
```

其中"-0"参数指定的是备份等级"-u"要求备份完毕之后将相应的信息存储到文件 /etc/dumpdates 留作记录

Linux uuencode命令

Linux uuencode命令用于将uuencode编码后的档案还原。

早期在许多 unix 系统的传送协定只能传送七位元字元，并不支援二进位档案，像中文文字档就有用到八位元，所以无法完整地送到另一架机器上。 uuencode 指令，可以将二进位档转换成七位元的档案，传送到另一架机器上再以 uudecode 还原。最常见的是用在以电子邮件传送二进位档。uuencode 编码后的资料都以 begin 开始，以 end 作为结束。

语法

```
compress[必要参数][选择参数][目录或者文件]
```

参数说明：

必要参数：

- 无

选择参数：

- h 显示帮助信息
- v 显示版本信息

实例

还原档案

```
# uuencode test.uud
```

Linux restore命令

Linux restore命令用来还原由dump操作所备份下来的文件或整个文件系统(一个分区)。

restore 指令所进行的操作和dump指令相反，dump操作可用来备份文件，而restore操作则是写回这些已备份的文件。

语法

```
restore [-cCvy][-b <区块大小>][-D <文件系统>][-f <备份文件>][-s <文件编号>] 或 restore [-chimvy
```

参数：

- -b<区块大小> 设置区块大小，单位是Byte。
- -c 不检查dump操作的备份格式，仅准许读取使用旧格式的备份文件。
- -C 使用对比模式，将备份的文件与现行的文件相互对比。
- -D<文件系统> 允许用户指定文件系统的名称。
- -f<备份文件> 从指定的文件中读取备份数据，进行还原操作。
- -h 仅解出目录而不包括与该目录相关的所有文件。
- -i 使用互动模式，在进行还原操作时，restore指令将依序询问用户。
- -m 解开符合指定的inode编号的文件或目录而非采用文件名称指定。
- -r 进行还原操作。
- -R 全面还原文件系统时，检查应从何处开始进行。
- -s<文件编号> 当备份数据超过一卷磁带时，您可以指定备份文件的编号。
- -t 指定文件名称，若该文件已存在备份文件中，则列出它们的名称。
- -v 显示指令执行过程。
- -x 设置文件名称，且从指定的存储媒体里读入它们，若该文件已存在在备份文件中，则将其还原到文件系统内。
- -y 不询问任何问题，一律以同意回答并继续执行指令。

Linux lha命令

Linux lha命令用于压缩或解压缩文件。

lha是从lharc演变而来的压缩程序，文件经它压缩后，会另外产生具有".lzh"扩展名的压缩文件。

语法

```
lha [-acdfglmnpqtuvx][-a <0/1/2>/u</0/1/2>][-<a/c/u>d][-<e/x>i][-<a/u>o][-<e/x>w=<目的目录
```

参数：

- -a或a 压缩文件，并加入到压缩文件内。
- -a<0/1/2>/u</0/1/2> 压缩文件时，采用不同的文件头。
- -c或c 压缩文件，重新建构新的压缩文件后，再将其加入。
- -d或d 从压缩文件内删除指定的文件。
- -<a/c/u>d或<a/c/u>d 压缩文件，然后将其加入，重新建构，更新压缩文件或，删除原始文件，也就是把文件移到压缩文件中。
- -e或e 解开压缩文件。
- -f或f 强制执行lha命令，在解压时会直接覆盖已有的文件而不加以询问。
- -g或g 使用通用的压缩格式，便于解决兼容性的问题。
- -<e/x>i或<e/x>i 解开压缩文件时，忽略保存在压缩文件内的文件路径，直接将其解压后存放在现行目录下或是指定的目录中。
- -l或l 列出压缩文件的相关信息。
- -m或m 此参数的效果和同时指定"-ad"参数相同。
- -n或n 不执行指令，仅列出实际执行会进行的动作。
- -<a/u>o或<a/u>o 采用lharc兼容格式，将压缩后的文件加入，更新压缩文件。
- -p或p 从压缩文件内输出到标准输出设备。
- -q或q 不显示指令执行过程。
- -t或t 检查备份文件内的每个文件是否正确无误。
- -u或u 更换较新的文件到压缩文件内。
- -u</0/1/2>或u</0/1/2> 在文件压缩时采用不同的文件头，然后更新到压缩文件内。
- -v或v 详细列出压缩文件的相关信息。
- -<e/x>w=<目的目录>或<e/x>w=<目的目录> 指定解压缩的目录。
- -x或x 解开压缩文件。
- -<a/u>z或<a/u>z 不压缩文件，直接把它加入，更新压缩文件。

实例

缩文件

```
# lha -a abc.lhz a.b //压缩a.b文件，压缩后生成 abc.lhz文件
```

压缩目录

```
# lha -a abc2 /home/hnlinux
```

解压文件到当前目录

```
# lha -xiw=agis abc  //解压文件abc
```

Linux gzip命令

Linux gzip命令用于压缩文件。

gzip是个使用广泛的压缩程序，文件经它压缩过后，其名称后面会多出".gz"的扩展名。

语法

```
gzip [-acdfhlLnNqrtvV][-S &lt;压缩字尾字符串&gt;][-&lt;压缩效率&gt;][--best/fast][文件...] 或
```

参数：

- -a或--ascii　使用ASCII文字模式。
- -c或--stdout或--to-stdout　把压缩后的文件输出到标准输出设备，不去更动原始文件。
- -d或--decompress或----uncompress　解开压缩文件。
- -f或--force　强行压缩文件。不理会文件名称或硬连接是否存在以及该文件是否为符号连接。
- -h或--help　在线帮助。
- -l或--list　列出压缩文件的相关信息。
- -L或--license　显示版本与版权信息。
- -n或--no-name　压缩文件时，不保存原来的文件名称及时间戳记。
- -N或--name　压缩文件时，保存原来的文件名称及时间戳记。
- -q或--quiet　不显示警告信息。
- -r或--recursive　递归处理，将指定目录下的所有文件及子目录一并处理。
- -S<压缩字尾字符串>或----suffix<压缩字尾字符串>　更改压缩字尾字符串。
- -t或--test　测试压缩文件是否正确无误。
- -v或--verbose　显示指令执行过程。
- -V或--version　显示版本信息。
- -<压缩效率>　压缩效率是一个介于1－9的数值，预设值为"6"，指定愈大的数值，压缩效率就会愈高。
- --best　此参数的效果和指定"-9"参数相同。
- --fast　此参数的效果和指定"-1"参数相同。

实例

压缩文件

```
[root@w3cschool.cc a]# ls //显示当前目录文件
a.c b.h d.cpp
[root@w3cschool.cc a]# gzip * //压缩目录下的所有文件
[root@w3cschool.cc a]# ls //显示当前目录文件
a.c.gz    b.h.gz    d.cpp.gz
[root@w3cschool.cc a]#
```

接范例1， 列出详细的信息

```
[root@w3cschool.cc a]# gzip -dv * //解压文件，并列出详细信息
a.c.gz:     0.0% -- replaced with a.c
b.h.gz:     0.0% -- replaced with b.h
d.cpp.gz:     0.0% -- replaced with d.cpp
[root@w3cschool.cc a]#
```

接范例1，显示压缩文件的信息

```
[root@w3cschool.cc a]# gzip -l *
     compressed    uncompressed ratio uncompressed_name
         24           0  0.0% a.c
         24           0  0.0% b.h
         26           0  0.0% d.cpp
```

Linux gzexe命令

Linux gzexe命令用于压缩执行文件。

gzexe是用来压缩执行文件的程序。当您去执行被压缩过的执行文件时，该文件会自动解压然后继续执行，和使用一般的执行文件相同。

语法

```
gzexe [-d][执行文件...]
```

参数：

- -d 解开压缩文件。

实例

压缩可执行文件

```
# gzexe abc
```

Linux zipinfo命令

Linux zipinfo命令用于列出压缩文件信息。

执行zipinfo指令可得知zip压缩文件的详细信息。

语法

```
zipinfo [-12hlmMstTvz][压缩文件][文件...][-x <范本样式>]
```

参数：

- -1 只列出文件名称。
- -2 此参数的效果和指定"-1"参数类似，但可搭配"-h","-t"和"-z"参数使用。
- -h 只列出压缩文件的文件名称。
- -l 此参数的效果和指定"-m"参数类似，但会列出原始文件的大小而非每个文件的压缩率。
- -m 此参数的效果和指定"-s"参数类似，但多会列出每个文件的压缩率。
- -M 若信息内容超过一个画面，则采用类似more指令的方式列出信息。
- -s 用类似执行"ls -l"指令的效果列出压缩文件内容。
- -t 只列出压缩文件内所包含的文件数目，压缩前后的文件大小及压缩率。
- -T 将压缩文件内每个文件的日期时间用年，月，日，时，分，秒的顺序列出。
- -v 详细显示压缩文件内每一个文件的信息。
- -x<范本样式> 不列出符合条件的文件的信息。
- -z 如果压缩文件内含有注释，就将注释显示出来。

实例

显示压缩文件信息

```
[root@w3cschool.cc a]# zipinfo cp.zip
Archive: cp.zip  486 bytes  4 files
-rw-r--r-- 2.3 unx    0 bx stor 24-May-10 18:54 a.c
-rw-r--r-- 2.3 unx    0 bx stor 24-May-10 18:54 b.c
-rw-r--r-- 2.3 unx    0 bx stor 24-May-10 18:54 c.c
-rw-r--r-- 2.3 unx    0 bx stor 24-May-10 18:54 e.c
4 files, 0 bytes uncompressed, 0 bytes compressed: 0.0%
[root@w3cschool.cc a]#
```

显示压缩文件中每个文件的信息

```
[root@w3cschool.cc a]# zipinfo -v cp.zip
Archive: cp.zip  486 bytes  4 files

End-of-central-directory record:
```

```
-------------------------------

  Actual offset of end-of-central-dir record:     464 (000001D0h)
  Expected offset of end-of-central-dir record:    464 (000001D0h)
  (based on the length of the central directory and its expected offset)

  This zipfile constitutes the sole disk of a single-part archive; its
  central directory contains 4 entries. The central directory is 248
  (000000F8h) bytes long, and its (expected) offset in bytes from the
  beginning of the zipfile is 216 (000000D8h).

  There is no zipfile comment.

Central directory entry #1:
---------------------------

  a.c

  offset of local header from start of archive:   0 (00000000h) bytes
  file system or operating system of origin:    Unix
  version of encoding software:           2.3
  minimum file system compatibility required:    MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   1.0
  compression method:                none (stored)
  file security status:               not encrypted
  extended local header:              no
  file last modified on (DOS date/time):      2010 May 24 18:54:26
  file last modified on (UT extra field modtime):  2010 May 24 18:54:26 local
  file last modified on (UT extra field modtime):  2010 May 24 10:54:26 UTC
  32-bit CRC value (hex):              00000000
  compressed size:                 0 bytes
  uncompressed size:                0 bytes
  length of filename:                3 characters
  length of extra field:              13 bytes
  length of file comment:              0 characters
  disk number on which file begins:         disk 1
  apparent file type:                binary
  Unix file attributes (100644 octal):       -rw-r--r--
  MS-DOS file attributes (00 hex):         none

  The central-directory extra field contains:
  - A subfield with ID 0x5455 (universal time) and 5 data bytes.
   The local extra field has UTC/GMT modification/access times.
  - A subfield with ID 0x7855 (Unix UID/GID) and 0 data bytes.

  There is no file comment.

Central directory entry #2:
---------------------------

  b.c

  offset of local header from start of archive:   54 (00000036h) bytes
  file system or operating system of origin:    Unix
  version of encoding software:           2.3
  minimum file system compatibility required:    MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   1.0
  compression method:                none (stored)
  file security status:               not encrypted
  extended local header:              no
  file last modified on (DOS date/time):      2010 May 24 18:54:26
  file last modified on (UT extra field modtime):  2010 May 24 18:54:26 local
  file last modified on (UT extra field modtime):  2010 May 24 10:54:26 UTC
  32-bit CRC value (hex):              00000000
  compressed size:                 0 bytes
  uncompressed size:                0 bytes
  length of filename:                3 characters
  length of extra field:              13 bytes
  length of file comment:              0 characters
  disk number on which file begins:         disk 1
  apparent file type:                binary
  Unix file attributes (100644 octal):       -rw-r--r--
```

```
  MS-DOS file attributes (00 hex):         none

  The central-directory extra field contains:
  - A subfield with ID 0x5455 (universal time) and 5 data bytes.
    The local extra field has UTC/GMT modification/access times.
  - A subfield with ID 0x7855 (Unix UID/GID) and 0 data bytes.

  There is no file comment.

Central directory entry #3:
---------------------------

  c.c

  offset of local header from start of archive:   108 (0000006Ch) bytes
  file system or operating system of origin:    Unix
  version of encoding software:           2.3
  minimum file system compatibility required:    MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   1.0
  compression method:                none (stored)
  file security status:               not encrypted
  extended local header:              no
  file last modified on (DOS date/time):      2010 May 24 18:54:26
  file last modified on (UT extra field modtime):  2010 May 24 18:54:26 local
  file last modified on (UT extra field modtime):  2010 May 24 10:54:26 UTC
  32-bit CRC value (hex):              00000000
  compressed size:                 0 bytes
  uncompressed size:                0 bytes
  length of filename:                3 characters
  length of extra field:              13 bytes
  length of file comment:              0 characters
  disk number on which file begins:         disk 1
  apparent file type:                binary
  Unix file attributes (100644 octal):       -rw-r--r--
  MS-DOS file attributes (00 hex):         none

  The central-directory extra field contains:
  - A subfield with ID 0x5455 (universal time) and 5 data bytes.
    The local extra field has UTC/GMT modification/access times.
  - A subfield with ID 0x7855 (Unix UID/GID) and 0 data bytes.

  There is no file comment.

Central directory entry #4:
---------------------------

  e.c

  offset of local header from start of archive:   162 (000000A2h) bytes
  file system or operating system of origin:    Unix
  version of encoding software:           2.3
  minimum file system compatibility required:    MS-DOS, OS/2 or NT FAT
  minimum software version required to extract:   1.0
  compression method:                none (stored)
  file security status:               not encrypted
  extended local header:              no
  file last modified on (DOS date/time):      2010 May 24 18:54:26
  file last modified on (UT extra field modtime):  2010 May 24 18:54:26 local
  file last modified on (UT extra field modtime):  2010 May 24 10:54:26 UTC
  32-bit CRC value (hex):              00000000
  compressed size:                 0 bytes
  uncompressed size:                0 bytes
  length of filename:                3 characters
  length of extra field:              13 bytes
  length of file comment:              0 characters
  disk number on which file begins:         disk 1
  apparent file type:                binary
  Unix file attributes (100644 octal):       -rw-r--r--
  MS-DOS file attributes (00 hex):         none

  The central-directory extra field contains:
  - A subfield with ID 0x5455 (universal time) and 5 data bytes.
```

```
 The local extra field has UTC/GMT modification/access times.
- A subfield with ID 0x7855 (Unix UID/GID) and 0 data bytes.

There is no file comment.
```

Linux zip命令

Linux zip命令用于压缩文件。

zip是个使用广泛的压缩程序，文件经它压缩后会另外产生具有".zip"扩展名的压缩文件。

语法

```
zip [-AcdDfFghjJKlLmoqrSTuvVwXyz$][-b <工作目录>][-ll][-n <字尾字符串>][-t <日期时间>][-<压缩效
```

参数：

- -A 调整可执行的自动解压缩文件。
- -b<工作目录> 指定暂时存放文件的目录。
- -c 替每个被压缩的文件加上注释。
- -d 从压缩文件内删除指定的文件。
- -D 压缩文件内不建立目录名称。
- -f 此参数的效果和指定"-u"参数类似，但不仅更新既有文件，如果某些文件原本不存在于压缩文件内，使用本参数会一并将其加入压缩文件中。
- -F 尝试修复已损坏的压缩文件。
- -g 将文件压缩后附加在既有的压缩文件之后，而非另行建立新的压缩文件。
- -h 在线帮助。
- -i<范本样式> 只压缩符合条件的文件。
- -j 只保存文件名称及其内容，而不存放任何目录名称。
- -J 删除压缩文件前面不必要的数据。
- -k 使用MS-DOS兼容格式的文件名称。
- -l 压缩文件时，把LF字符置换成LF+CR字符。
- -ll 压缩文件时，把LF+CR字符置换成LF字符。
- -L 显示版权信息。
- -m 将文件压缩并加入压缩文件后，删除原始文件，即把文件移到压缩文件中。
- -n<字尾字符串> 不压缩具有特定字尾字符串的文件。
- -o 以压缩文件内拥有最新更改时间的文件为准，将压缩文件的更改时间设成和该文件相同。
- -q 不显示指令执行过程。
- -r 递归处理，将指定目录下的所有文件和子目录一并处理。
- -S 包含系统和隐藏文件。
- -t<日期时间> 把压缩文件的日期设成指定的日期。
- -T 检查备份文件内的每个文件是否正确无误。

- -u 更换较新的文件到压缩文件内。
- -v 显示指令执行过程或显示版本信息。
- -V 保存VMS操作系统的文件属性。
- -w 在文件名称里假如版本编号，本参数仅在VMS操作系统下有效。
- -x<范本样式> 压缩时排除符合条件的文件。
- -X 不保存额外的文件属性。
- -y 直接保存符号连接，而非该连接所指向的文件，本参数仅在UNIX之类的系统下有效。
- -z 替压缩文件加上注释。
- -$ 保存第一个被压缩文件所在磁盘的卷册名称。
- -<压缩效率> 压缩效率是一个介于1-9的数值。

实例

压缩文件

```
# zip -v cp.zip a.c b.c c.c e.c
 adding: a.c    (in=0) (out=0) (stored 0%)
 adding: b.c    (in=0) (out=0) (stored 0%)
 adding: c.c    (in=0) (out=0) (stored 0%)
 adding: e.c    (in=0) (out=0) (stored 0%)
total bytes=0, compressed=0 -> 0% savings
```

压缩文件

```
# [root@ubuntu a]# zip -v cp2.zip *

#
```

压缩目录

```
# zip -r cp3.zip /root/
```

从压缩文件中删除文件

```
# zip -dv cp.zip a.c
```

Linux unzip命令

Linux unzip命令用于解压缩zip文件

unzip为.zip压缩文件的解压缩程序。

语法

```
unzip [-cflptuvz][-agCjLMnoqsVX][-P <密码>][.zip文件][文件][-d <目录>][-x <文件>] 或 unzip [-
```

参数：

- -c 将解压缩的结果显示到屏幕上，并对字符做适当的转换。
- -f 更新现有的文件。
- -l 显示压缩文件内所包含的文件。
- -p 与-c参数类似，会将解压缩的结果显示到屏幕上，但不会执行任何的转换。
- -t 检查压缩文件是否正确。
- -u 与-f参数类似，但是除了更新现有的文件外，也会将压缩文件中的其他文件解压缩到目录中。
- -v 执行是时显示详细的信息。
- -z 仅显示压缩文件的备注文字。
- -a 对文本文件进行必要的字符转换。
- -b 不要对文本文件进行字符转换。
- -C 压缩文件中的文件名称区分大小写。
- -j 不处理压缩文件中原有的目录路径。
- -L 将压缩文件中的全部文件名改为小写。
- -M 将输出结果送到more程序处理。
- -n 解压缩时不要覆盖原有的文件。
- -o 不必先询问用户，unzip执行后覆盖原有文件。
- -P<密码> 使用zip的密码选项。
- -q 执行时不显示任何信息。
- -s 将文件名中的空白字符转换为底线字符。
- -V 保留VMS的文件版本信息。
- -X 解压缩时同时回存文件原来的UID/GID。
- [.zip文件] 指定.zip压缩文件。
- [文件] 指定要处理.zip压缩文件中的哪些文件。
- -d<目录> 指定文件解压缩后所要存储的目录。
- -x<文件> 指定不要处理.zip压缩文件中的哪些文件。

- -Z unzip -Z等于执行zipinfo指令。

实例

显示压缩文件信息

```
# unzip -l abc.zip
Archive: abc.zip
 Length   Date  Time  Name
-------- ----  ----  ----
  94618 05-21-10 20:44  a11.jpg
  202001 05-21-10 20:44  a22.jpg
    16 05-22-10 15:01  11.txt
  46468 05-23-10 10:30  w456.JPG
  140085 03-14-10 21:49  my.asp
--------          -------
  483188          5 files
```

解压文件

```
# unzip -v abc.zip
Archive: abc.zip
Length  Method  Size Ratio  Date  Time  CRC-32  Name
-------- ------ ------- -----  ----  ----  ------  ----
  94618 Defl:N  93353  1% 05-21-10 20:44 9e661437 a11.jpg
 202001 Defl:N  201833  0% 05-21-10 20:44 1da462eb a22.jpg
   16 Stored    16  0% 05-22-10 15:01 ae8a9910 ? +-|￥+-? (11).txt
  46468 Defl:N  39997 14% 05-23-10 10:30 962861f2 w456.JPG
 140085 Defl:N  36765 74% 03-14-10 21:49 836fcc3f my.asp
--------     ------- ---            -------
 483188      371964 23%              5 files
```

Linux uudecode命令

Linuxuudecode 将 uuencode 编码后的档案还原， uudecode 只会将 begin 与 end 标记之间的编码资料还原，程序会跳过标记以外的资料。

语法

```
uuencode [-hv] [file1 ...]</p>
```

参数：

- h 列出指令使用格式 (help)
- v 列出版本讯息

实例

将 file.uud 还原，而还原后的档名储存在 file.uud 档中。

```
uuencode file.uud
```

可以一起还原好几个档案。

```
uuencode file1.uud file2.uud
```

Linux tar命令

Linux tar命令用于备份文件。

tar是用来建立，还原备份文件的工具程序，它可以加入，解开备份文件内的文件。

语法

```
tar [-ABcdgGhiklmMoOpPrRsStuUvwWxzZ][-b <区块数目>][-C <目的目录>][-f <备份文件>][-F <Script文
```

参数：

- -A或--catenate 新增温暖件到已存在的备份文件。
- -b<区块数目>或--blocking-factor=<区块数目> 设置每笔记录的区块数目，每个区块大小为12Bytes。
- -B或--read-full-records 读取数据时重设区块大小。
- -c或--create 建立新的备份文件。
- -C<目的目录>或--directory=<目的目录> 切换到指定的目录。
- -d或--diff或--compare 对比备份文件内和文件系统上的文件的差异。
- -f<备份文件>或--file=<备份文件> 指定备份文件。
- -F<Script文件>或--info-script=<Script文件> 每次更换磁带时，就执行指定的Script文件。
- -g或--listed-incremental 处理GNU格式的大量备份。
- -G或--incremental 处理旧的GNU格式的大量备份。
- -h或--dereference 不建立符号连接，直接复制该连接所指向的原始文件。
- -i或--ignore-zeros 忽略备份文件中的0 Byte区块，也就是EOF。
- -k或--keep-old-files 解开备份文件时，不覆盖已有的文件。
- -K<文件>或--starting-file=<文件> 从指定的文件开始还原。
- -l或--one-file-system 复制的文件或目录存放的文件系统，必须与tar指令执行时所处的文件系统相同，否则不予复制。
- -L<媒体容量>或-tape-length=<媒体容量> 设置存放每体的容量，单位以1024 Bytes计算。
- -m或--modification-time 还原文件时，不变更文件的更改时间。
- -M或--multi-volume 在建立，还原备份文件或列出其中的内容时，采用多卷册模式。
- -N<日期格式>或--newer=<日期时间> 只将较指定日期更新的文件保存到备份文件里。
- -o或--old-archive或--portability 将资料写入备份文件时使用V7格式。
- -O或--stdout 把从备份文件里还原的文件输出到标准输出设备。
- -p或--same-permissions 用原来的文件权限还原文件。
- -P或--absolute-names 文件名使用绝对名称，不移除文件名称前的"/"号。

- -r或--append 新增文件到已存在的备份文件的结尾部分。
- -R或--block-number 列出每个信息在备份文件中的区块编号。
- -s或--same-order 还原文件的顺序和备份文件内的存放顺序相同。
- -S或--sparse 倘若一个文件内含大量的连续0字节，则将此文件存成稀疏文件。
- -t或--list 列出备份文件的内容。
- -T<范本文件>或--files-from=<范本文件> 指定范本文件，其内含有一个或多个范本样式，让tar解开或建立符合设置条件的文件。
- -u或--update 仅置换较备份文件内的文件更新的文件。
- -U或--unlink-first 解开压缩文件还原文件之前，先解除文件的连接。
- -v或--verbose 显示指令执行过程。
- -V<卷册名称>或--label=<卷册名称> 建立使用指定的卷册名称的备份文件。
- -w或--interactive 遭遇问题时先询问用户。
- -W或--verify 写入备份文件后，确认文件正确无误。
- -x或--extract或--get 从备份文件中还原文件。
- -X<范本文件>或--exclude-from=<范本文件> 指定范本文件，其内含有一个或多个范本样式，让ar排除符合设置条件的文件。
- -z或--gzip或--ungzip 通过gzip指令处理备份文件。
- -Z或--compress或--uncompress 通过compress指令处理备份文件。
- -<设备编号><存储密度> 设置备份用的外围设备编号及存放数据的密度。
- --after-date=<日期时间> 此参数的效果和指定"-N"参数相同。
- --atime-preserve 不变更文件的存取时间。
- --backup=<备份方式>或--backup 移除文件前先进行备份。
- --checkpoint 读取备份文件时列出目录名称。
- --concatenate 此参数的效果和指定"-A"参数相同。
- --confirmation 此参数的效果和指定"-w"参数相同。
- --delete 从备份文件中删除指定的文件。
- --exclude=<范本样式> 排除符合范本样式的问家。
- --group=<群组名称> 把加入设备文件中的文件的所属群组设成指定的群组。
- --help 在线帮助。
- --ignore-failed-read 忽略数据读取错误，不中断程序的执行。
- --new-volume-script=<Script文件> 此参数的效果和指定"-F"参数相同。
- --newer-mtime 只保存更改过的文件。
- --no-recursion 不做递归处理，也就是指定目录下的所有文件及子目录不予处理。
- --null 从null设备读取文件名称。
- --numeric-owner 以用户识别码及群组识别码取代用户名称和群组名称。
- --owner=<用户名称> 把加入备份文件中的文件的拥有者设成指定的用户。
- --posix 将数据写入备份文件时使用POSIX格式。
- --preserve 此参数的效果和指定"-ps"参数相同。
- --preserve-order 此参数的效果和指定"-A"参数相同。
- --preserve-permissions 此参数的效果和指定"-p"参数相同。

- --record-size=<区块数目> 此参数的效果和指定"-b"参数相同。
- --recursive-unlink 解开压缩文件还原目录之前，先解除整个目录下所有文件的连接。
- --remove-files 文件加入备份文件后，就将其删除。
- --rsh-command=<执行指令> 设置要在远端主机上执行的指令，以取代rsh指令。
- --same-owner 尝试以相同的文件拥有者还原问家你。
- --suffix=<备份字尾字符串> 移除文件前先行备份。
- --totals 备份文件建立后，列出文件大小。
- --use-compress-program=<执行指令> 通过指定的指令处理备份文件。
- --version 显示版本信息。
- --volno-file=<编号文件> 使用指定文件内的编号取代预设的卷册编号。

实例

压缩文件 非打包

```
# touch a.c
# tar -czvf test.tar.gz a.c   //压缩 a.c文件为test.tar.gz
a.c
```

列出压缩文件内容

```
# tar -tzvf test.tar.gz
-rw-r--r-- root/root     0 2010-05-24 16:51:59 a.c
```

解压文件

tar -xzvf test.tar.gz a.c

Linux命令大全 - 设备管理

setleds	loadkeys	rdev	dumpkeys
MAKEDEV			

Linux setleds命令

Linux setleds命令用来设定键盘上方三个 LED 的状态。在 Linux 中，每一个虚拟主控台都有独立的设定。

语法

```
setleds [-v] [-L] [-D] [-F] [{+|-}num] [{+|-}caps] [{+|-}scroll]
```

参数：

- -F：预设的选项，设定虚拟主控台的状态。
- -D：除了改变虚拟主控台的状态外，还改变预设的状态。
- -L：不改变虚拟主控台的状态，但直接改变 LED 显示的状态。这会使得 LDE 显示和目前虚拟主控台的状态不符合。我们可以在稍后用 -L 且不含其它选项的 setleds 命令回复正常状态。
- -num +num：将数字键打开或关闭。
- -caps +caps：把大小写键打开或关闭。
- -scroll +scroll：把选项键打开或关闭。

实例

将数字键打开，其馀二个灯关闭。

```
# setleds +num -caps -scroll
```

Linux loadkeys命令

Linux loadkeys命令可以根据一个键盘定义表改变 linux 键盘驱动程序转译键盘输入过程。详细的说明请参考 dumpkeys。

语法

```
loadkeys [ -d --default ] [ -h --help ] [ -q --quiet ] [ -v --verbose [ -v --verbose ]...
```

参数:

- -v --verbose：印出详细的资料，你可以重复以增加详细度。
- -q --quiet：不要显示任何讯息。
- -c --clearcompose：清除所有 composite 定义。
- -s --clearstrings：将定串定义表清除。

实例

```
定义按键组合
<pre>
# loadkeys
control alt keycode 88 = F80 //现确定键代码
string F80="w3cschool.cc" //给变变量设定值
//按下 Ctrl + D键 确定输入

//效果：按下 Ctrl +Alt + F12 输出 Lx138.Com

# dumpkeys --funcs-only //显示功能键

……省略部分结果
string F3 = "\033[[C"
string F4 = "\033[[D"
string F5 = "\033[[E"
string F6 = "\033[17~"
string F7 = "\033[18~"
string F8 = "\033[19~"
string F9 = "\033[20~"
string F10 = "\033[21~"
string F11 = "\033[23~"
string F12 = "\033[24~"
string F13 = "\033[25~"
string F14 = "\033[26~"
string F15 = "\033[28~"
string F16 = "\033[29~"
string F17 = "\033[31~"
string F18 = "\033[32~"
string F19 = "\033[33~"
string F20 = "\033[34~"
string Find = "\033[1~"
string Insert = "\033[2~"
string Remove = "\033[3~"
string Select = "\033[4~"
string Prior = "\033[5~"
string Next = "\033[6~"
string Macro = "\033[M"
string Pause = "\033[P"
string F80 = "w3cschool.cc"
```

Linux rdev命令

Linux rdev命令可以用来查询/设置内核映像文件的根设备，RAM 磁盘大小或视频模式。

不带任何参数的 rdev 命令将输出当前根文件系统的 /etc/mtab 文件行。不带任何参数的 ramsize, vidmode, 和 rootflags 将显示帮助信息。

语法

```
rdev [-rsvh ] [-o offset ] [ image [value [ offset ] ] ]</p>
```

但是随著使用者想要设定的参数的不同，底下的方式也是一样：

```
rdev [ -o offset ] [ image [ root_device [ offset ] ] ]
```

```
swapdev [ -o offset ] [ image [ swap_device [ offset ] ] ]
```

```
ramsize [ -o offset ] [ image [ size [ offset ] ] ]
```

```
videomode [ -o offset ] [ image [ mode [ offset ] ] ]
```

```
rootflags [ -o offset ] [ image [ flags [ offset ] ] ]
```

参数：

- -r： 使得 rdev 作为 ramsize 运行。
- -R： 使得 rdev 作为 rootflags 运行。
- -v： 使得 rdev 作为 vidmode 运行。
- -h： 提供帮助。

Linux dumpkeys命令

Linux dumpkeys命令用于显示键盘映射表，输出的内容可以被loadkeys命令识别,改变映射关系。

语法

```
dumpkey[选择参数]
```

参数说明:

- -i 驱动信息(键码范围、数量、状态键)
- -l 详细驱动信息
- -n 十六进制显示
- -f 显示全部信息
- -1 分行显示按键组合
- -S 设定输出格式(0：预设 1：完整 2：分行 3简单)
- --funcs-only 功能键信息
- --keys-only 键组合信息
- --compose-only 普通键信息

实例

显示功能键信息

```
# dumpkeys --funcs-only
string F1 = "\033[[A"
string F2 = "\033[[B"
string F3 = "\033[[C"
string F4 = "\033[[D"
string F5 = "\033[[E"
string F6 = "\033[17~"
string F7 = "\033[18~"
string F8 = "\033[19~"
string F9 = "\033[20~"
string F10 = "\033[21~"
string F11 = "\033[23~"
string F12 = "\033[24~"
string F13 = "\033[25~"
string F14 = "\033[26~"
string F15 = "\033[28~"
string F16 = "\033[29~"
string F17 = "\033[31~"
string F18 = "\033[32~"
string F19 = "\033[33~"
string F20 = "\033[34~"
string Find = "\033[1~"
string Insert = "\033[2~"
string Remove = "\033[3~"
string Select = "\033[4~"
string Prior = "\033[5~"
string Next = "\033[6~"
string Macro = "\033[M"
string Pause = "\033[P"
root@snail-hnlinux:~#
```

显示驱动信息

```
# dumpkeys -i
键值码范围被内核支持: 1 - 255
可绑定到键值的动作最大值: 256
实际使用的键值数: 128
其中 121 已动态分配
被内核支持的动作码值范围
0x0000 - 0x00ff
0x0100 - 0x01ff
0x0200 - 0x0213
0x0300 - 0x0313
0x0400 - 0x0405
0x0500 - 0x05ff
0x0600 - 0x0603
0x0700 - 0x0708
0x0800 - 0x08ff
0x0900 - 0x0919
0x0a00 - 0x0a08
0x0b00 - 0x0bff
0x0c00 - 0x0c08
0x0d00 - 0x0dff
0x0e00 - 0x0e0a
内核支持的功能键数：256
编写定义的最大nr: 256
实际使用的编写定义nr: 68
```

Linux MAKEDEV命令

Linux MAKEDEV命令用于新增 /dev/ 下的装置档案，多数分区已经将所有的档案都产生，故一般而言不太会需要用到这个命令。

语法

```
MAKEDEV -V
MAKEDEV [ -n ] [ -v ] update
MAKEDEV [ -n ] [ -v ] [ -d ] device ...
```

Makefile

makefile定义了一系列的规则来指定，哪些文件需要先编译，哪些文件需要后编译，哪些文件需要重新编译，甚至于进行更复杂的功能操作，因为 makefile就像一个Shell脚本一样，其中也可以执行操作系统的命令。 Make工具最主要也是最基本的功能就是通过makefile文件来描述源程序之间的相互关系并自动维护编译工作。而makefile 文件需要按照某种语法进行编写，文件中需要说明如何编译各个源文件并连接生成可执行文件，并要求定义源文件之间的依赖关系。makefile 文件是许多编译器--包括 Windows NT 下的编译器--维护编译信息的常用方法，只是在集成开发环境中，用户通过友好的界面修改 makefile 文件而已。在 UNIX 系统中，习惯使用 Makefile 作为 makefile 文件。如果要使用其他文件作为 makefile，则可利用类似下面的 make 命令选项指定 makefile 文件。 一个文件，指示程序如何编译和链接程序。 makefile文件的默认名称是名副其实的Makefile，但可以指定一个命令行选项的名称。 make 程序有助于您在开发大型程序跟踪整个程序，其中部分已经改变，只有那些编译自上次编译的程序，它已经改变了部分。

关于编译阶段编译一个小的C程序至少需要一个单一的文件.h文件（如适用）。虽然命令执行此任务只需CC file.c中，有3个步骤，以取得最终的可执行程序，如下所示：编译阶段：所有的C语言代码.c文件中被转换成一个低级语言汇编语言;决策.s文件。汇编阶段：前阶段所作的汇编语言代码，然后转换成目标代码的代码片段，该计算机直接理解。目标代码文件.o 结束。链接阶段：编译程序涉及到链接的对象代码的代码库，其中包含一定的“内置”的功能，如printf的最后阶段。这个阶段产生一个可执行程序，默认情况下，这是名为a.out。

为什么需要Makefile？ - Makefile

对于本次教程中的讨论，假定有以下的源文件。

- main.cpp
- hello.cpp
- factorial.cpp
- functions.h

main.cpp 文件的内容

```
#include &lt;iostream.h&gt;

#include "functions.h"

int main(){
    print_hello();
    cout &lt;&lt; endl;
    cout &lt;&lt; "The factorial of 5 is " &lt;&lt; factorial(5) &lt;&lt; endl;
    return 0;
}
```

hello.cpp 文件的内容

```
#include &lt;iostream.h&gt;

#include "functions.h"

void print_hello(){
   cout &lt;&lt; "Hello World!";
}
```

factorial.cpp 文件的内容

```
#include "functions.h"

int factorial(int n){
    if(n!=1){
    return(n * factorial(n-1));
    }
    else return 1;
}
```

functions.h 内容

```
void print_hello();
int factorial(int n);
```

琐碎的方法来编译的文件，并获得一个可执行文件，通过运行以下命令：

```
CC  main.cpp hello.cpp factorial.cpp -o hello
```

这上面的命令将生成二进制的Hello。在我们的例子中，我们只有四个文件，我们知道的函数调用序列，因此它可能是可行的，上面写的命令的手，准备最后的二进制。但对于大的项目，我们将有源代码文件成千上万的文件，就很难保持二进制版本。

make命令允许您管理大型程序或程序组。当开始编写较大的程序，你会发现，重新编译较大的程序，需要更长的时间比重新编译的短节目。此外会发现通常只能在一小部分的程序（如单一功能正在调试），其余的程序不变。

在随后的章节中，我们将看到项目是如何准备一个makefile。

Makefile 宏 - Makefile

make程序允许您使用宏，这是类似的变量。 = 一对一个Makefile中定义的宏。例如：

```
MACROS=  -me
PSROFF=  groff -Tps
DITROFF= groff -Tdvi
CFLAGS= -O -systype bsd43
LIBS = "-lncurses -lm -lsdl"
MYFACE = ":*)"
```

- 特殊的宏

目标规则集发出任何命令之前，有一些特殊的预定义宏。

- $@ 到的文件的名称。
- $? 是改变的眷属的名字。

因此，举例来说，我们可以使用一个规则

```
hello: main.cpp hello.cpp factorial.cpp
    $(CC) $(CFLAGS) $? $(LDFLAGS) -o $@

alternatively:

hello: main.cpp hello.cpp factorial.cpp
        $(CC) $(CFLAGS) $@.cpp $(LDFLAGS) -o $@
```

在这个例子中$@代表 hello， $? 或$@.cpp将拾取所有更改的源文件。

有两个比较特殊的隐含规则中使用的宏。它们是

- $< 导致该操作的相应的文件中的名称。
- $* 前缀共享目标和相关文件。

常见的隐含规则的构造 .o（对象）文件，.cpp（源文件）。

```
.o.cpp:
        $(CC) $(CFLAGS) -c $&lt;

alternatively

.o.cpp:
        $(CC) $(CFLAGS) -c $*.c
```

注意：要获得更详细的关于Makefile规则，在另一部分。

- 传统宏

有很多默认的宏（输入“make -p”打印出来的默认值）。大多数使用它们的规则是很明显的：

这些预定义变量，即。在隐含规则中使用的宏分为两大类：那些程序名（例如CC）和那些含有参数的程序（如CFLAGS）。

下面是一些比较常见的变量用作内置规则：makefile文件的程序名称的表。

AR	Archive-maintaining program; default `ar'.
AS	Program for compiling assembly files; default `as'.
CC	Program for compiling C programs; default `cc'.
CO	Program for checking out files from RCS; default `co'.
CXX	Program for compiling C++ programs; default `g++'.
CPP	Program for running the C preprocessor, with results to standard output; default `$(CC) -E'.
FC	Program for compiling or preprocessing Fortran and Ratfor programs; default `f77'.
GET	Program for extracting a file from SCCS; default `get'.
LEX	Program to use to turn Lex grammars into source code; default `lex'.
YACC	Program to use to turn Yacc grammars into source code; default `yacc'.
LINT	Program to use to run lint on source code; default `lint'.
M2C	Program to use to compile Modula-2 source code; default `m2c'.
PC	Program for compiling Pascal programs; default `pc'.
MAKEINFO	Program to convert a Texinfo source file into an Info file; default `makeinfo'.
TEX	Program to make TeX dvi files from TeX source; default `tex'.
TEXI2DVI	Program to make TeX dvi files from Texinfo source; default `texi2dvi'.
WEAVE	Program to translate Web into TeX; default `weave'.
CWEAVE	Program to translate C Web into TeX; default `cweave'.
TANGLE	Program to translate Web into Pascal; default `tangle'.
CTANGLE	Program to translate C Web into C; default `ctangle'.
RM	Command to remove a file; default `rm -f'.

这里是一个变量，其值是上述程序的额外的参数表。所有这些的默认值是空字符串，除非另有说明。

ARFLAGS	Flags to give the archive-maintaining program; default `rv'.
ASFLAGS	Extra flags to give to the assembler (when explicitly invoked on a `.s' or` .S' file).
CFLAGS	Extra flags to give to the C compiler.
CXXFLAGS	Extra flags to give to the C compiler.
COFLAGS	Extra flags to give to the RCS co program.
CPPFLAGS	Extra flags to give to the C preprocessor and programs that use it (the C and Fortran compilers).
FFLAGS	Extra flags to give to the Fortran compiler.
GFLAGS	Extra flags to give to the SCCS get program.
LDFLAGS	Extra flags to give to compilers when they are supposed to invoke the linker, `ld'.
LFLAGS	Extra flags to give to Lex.
YFLAGS	Extra flags to give to Yacc.
PFLAGS	Extra flags to give to the Pascal compiler.
RFLAGS	Extra flags to give to the Fortran compiler for Ratfor programs.
LINTFLAGS	Extra flags to give to lint.

注：您可以取消 `-R'或` --no-builtin-variables“选项隐含规则使用的所有变量。

如，也可以在命令行中定义的宏

```
make CPP = /home/courses/cop4530/spring02
```

Makefile 定义依赖性 - Makefile

这是很常见的，最终的二进制文件将依赖于各种源代码和源代码的头文件。依存关系是重要的，因为他们告诉对任何目标的源。请看下面的例子

```
hello: main.o factorial.o hello.o
    $(CC) main.o factorial.o hello.o -o hello
```

在这里，我们告诉hello 依赖main.o，factorial.o和hello.o，所以每当有任何变化，这些目标文件将采取行动。

同时我们会告诉如何准备 .o文件，所以我们必须定义这些依赖也如下

```
main.o: main.cpp functions.h
    $(CC) -c main.cpp

factorial.o: factorial.cpp functions.h
    $(CC) -c factorial.cpp

hello.o: hello.cpp functions.h
    $(CC) -c hello.cpp
```

Makefile 定义规则 - Makefile

一个Makefile目标规则的一般语法

```
target [target...] : [dependent ....]
[ command ...]
```

方括号中的项是可选的，省略号是指一个或多个。注意标签，每个命令前需要。

下面给出一个简单的例子，定义了一个规则使您的目标从 hello 其他三个文件。 ``` hello: main.o factorial.o hello.o $(CC) main.o factorial.o hello.o -o hello ``` 注：在这个例子中，你必须放弃规则，使所有对象从源文件的文件 语义是相当简单的。当"make target"发现目标规则适用，如有眷属的新目标，使执行的命令一次一个（后宏替换）。如果有任何依赖进行，即先发生（让您拥有一个递归）。 如果有任何命令返回一个失败状态，MAKE将终止。这就是为什么看到规则，如： ``` clean: -rm *.o *~ core paper ``` Make忽略一个破折号开头的命令行返回的状态。例如。如果没有核心文件，谁在乎呢？ Make 会 echo 宏字符串替换的命令后，告诉发生了什么事，因为它发生。有时可能想要把它们关掉。例如： ``` install: @echo You must be root to install ``` 大家所期望的Makefile的 某些目标。应该总是先浏览，但它的合理预期的目标（或只是做），安装，清洁，会发现。 * make all - 编译一切，让你可以在本地测试，之前安装的东西。 * make install - 应安装在正确的地方的东西。但看出来的东西都安装在正确的地方为系统。 * make clean - 应该清理的东西。摆脱的可执行文件，任何临时文件，目标文件等。 ## Makefile的隐含规则 该命令应该在所有情况下，我们建立一个可执行x的的源代码x.cpp的作为一个隐含的规则，这可以说： ``` .cpp: $(CC) $(CFLAGS) $@.cpp $(LDFLAGS) -o $@ ``` 这种隐含的规则说，如何make c, x.c- 运行x.c 调用输出x。规则是隐式的，因为没有特定的目标提到。它可用于在所有的情况下。 另一种常见的隐含规则的构造 .o（对象）文件和 .cpp （源文件）。 ``` .o.cpp: $(CC) $(CFLAGS) -c >

Makefile 自定义后缀规则 - Makefile

就其本身而言，make已经知道，为了创建一个 .o文件，就必须使用 cc-c 相应的c文件。 建成MAKE这些规则，可以利用这一点来缩短Makefile。如果仅仅只是表示 .h 文件的 Makefile依赖线，依赖于目前的目标是，MAKE会知道，相应的文件已规定。你甚至不需要编译器包括命令。

这减少了我们的Makefile更多，如下所示：

```
OBJECTS = main.o hello.o factorial.o
hello: $(OBJECTS)
        cc $(OBJECTS) -o hello
hellp.o: functions.h
main.o: functions.h
factorial.o: functions.h
```

Make 使用一个特殊的目标，故名 .SUFFIXES允许你定义自己的后缀。例如，依赖线：

```
.SUFFIXES: .foo .bar
```

告诉make， 将使用这些特殊的后缀，以使自己的规则。

如何让 make 已经知道如何从 .c 文件生成 .o文件。类似的可以定义规则以下列方式：

```
.foo.bar:
        tr '[A-Z][a-z]' '[N-Z][A-M][n-z][a-m]' &lt; $&lt; &gt; $@
.c.o:
        $(CC) $(CFLAGS) -c $&lt;
```

第一条规则允许你创建一个 .bar 文件从 .foo文件。 （不要担心它做什么，它基本上打乱文件）第二条规则 .c文件创建一个 .o 文件中使用的默认规则。

Makefile 指令 - Makefile

有好几种指令以不同的形式。让程序可能不支持所有指令。因此，请检查make是否支持指令，我们这里解释。 GNU make支持这些指令

条件指令

条件的指令

- ifeq 指令开始的条件，指定的条件。它包含两个参数，用逗号分隔，并用括号括起。两个参数进行变量替换，然后对它们进行比较。该行的makefile继IFEQ的服从如果两个参数的匹配，否则会被忽略。
- ifneq 指令开始的条件，指定的条件。它包含两个参数，用逗号分隔，并用括号括起。两个参数进行变量替换，然后对它们进行比较。makefile ifneq 遵守如果两个参数不匹配，否则会被忽略。
- ifdef 指令开始的条件，指定的条件。它包含单参数。如果给定的参数为真，则条件为真。
- ifndef 指令开始的条件，指定的条件。它包含单参数。如果给定的是假的，那么条件为真。
- else 指令会导致以下行如果前面的条件未能被遵守。在上面的例子中，这意味着第二个选择连接命令时使用的第一种选择是不使用。它是可选的，在有条件有一个else。
- endif 指令结束条件。每一个条件必须与endif结束。

条件式指令的语法

一个简单的条件，没有其他的语法如下:

```
conditional-directive
text-if-true
endif
```

文本如果真可以是任何行文字，被视为makefile文件的一部分，如果条件为真。如果条件是假的，没有文字来代替。

一个复杂的语法条件如下：

```
conditional-directive
text-if-true
else
text-if-false
endif
```

如果条件为真时，文本，如果真正的使用，否则，如果假文本来代替。的文本，如果错误的数量可以是任意的文本行。

有条件的指令的语法是相同的，无论是简单或复杂的条件。有四种不同的测试不同条件下的指令。这里是一个表：

```
ifeq (arg1, arg2)
ifeq 'arg1' 'arg2'
ifeq "arg1" "arg2"
ifeq "arg1" 'arg2'
ifeq 'arg1' "arg2"
```

上述条件相反的指令如下

```
ifneq (arg1, arg2)
ifneq 'arg1' 'arg2'
ifneq "arg1" "arg2"
ifneq "arg1" 'arg2'
ifneq 'arg1' "arg2"
```

条件式指令示例

```
libs_for_gcc = -lgnu
normal_libs =

foo: $(objects)
ifeq ($(CC),gcc)
        $(CC) -o foo $(objects) $(libs_for_gcc)
else
        $(CC) -o foo $(objects) $(normal_libs)
endif
```

include 指令

include指令告诉make暂停读取当前makefile文件和读取一个或多个其它的makefile，然后再继续。该指令是一行在makefile中，看起来像这样：

```
include filenames...
```

文件名可以包含shell文件名模式。允许额外的空格开头的行被忽略，但不允许一个标签。例如，如果有三个 `.mk',` .mk' files, `a.mk',` b.mk', and `c.mk', and $(bar) 扩展到bash中，然后下面的表达式。

```
include foo *.mk $(bar)

is equivalent to

include foo a.mk b.mk c.mk bish bash
```

当MAKE处理包括指令，它包含的makefile暂停读取，并从各列出文件中依次读取。当这个过程完成，使读取指令出现在其中的makefile的恢复。

override 指令

如果一个变量已经设置的命令参数，在makefile中被忽略的普通任务。如果要设置makefile的变量，即使它被设置的命令参数，可以使用一个override指令，这是一行看起来像这样：

```
override variable = value

or

override variable := value
```

Makefile 文件重新编译 - Makefile

make 程序是一个智能的实用程序和工作根据在源文件中的变化。如果有四个文件main.cpp，hello.cpp，factorial.cpp和functions.h。这里所有reamining文件是依赖functions.h，main.cpp的是依赖于hello.cpp，factorical.cpp。因此，如果做任何改变functions.h然后将重新编译所有源文件来生成新的对象文件。但是，如果做任何改变main.cpp，因为这是不依赖任何其他的过滤，那么在这种情况下，只有main.cpp文件将被重新编译和hellp.cpp factorial.cpp将无法重新编译。

虽然编译一个文件时，MAKE检查目标文件和比较时间表带，如果源文件有更新的时间戳比目标文件，然后将生成新的对象文件，假设源文件已被改变。

避免重新编译

有可能是项目包括成千上万的文件。有时候可能已经改变了一个源文件，但不想重新编译所有依赖于它的文件。例如，假设添加宏到一个头文件或声明，许多其他文件依赖。假设在头文件中的任何变化需要重新编译所有相关文件，但要知道，他们并不需要重新编译，你宁可不要浪费时间等待他们的编译。

如果预期改变头文件的问题之前，可以使用`-t'标志位。这个标志告诉make命令不运行的规则，而是来标记目标，迄今为止，通过改变它的最后修改日期。遵循以下步骤：

1. 使用命令'make'来重新编译真的需要重新编译源文件。
2. 在头文件中进行更改。
3. 使用命令`-t'来纪念所有的目标文件为最新。下一次运行make，在头文件中的变化不会引起任何重新编译。

如果已经改变了头文件的时候，有一些文件就需要重新编译，做到这一点已经太晚了。相反，可以使用`-o文件“的标志，这标志着一个指定的文件作为”old“。这意味着该文件本身不会被重制并没有别的其交代将被重制。遵循以下步骤：

1. 重新编译源文件，需要编制独立的特定头文件的原因， `make -o headerfile'。如果涉及几个头文件，使用一个单独的` -o'选项，每个头文件。
2. 轻触所有目标文件使用`make -t'.

Makefile 其他功能 - Makefile

make 递归使用

递归使用的手段使用，make在makefile作为命令。这种技术是非常有用的，当你想要的makefile各种子系统组成一个更大的系统。例如，假设你有一个子目录，子目录都有其自己的makefile，并且您希望所在目录的makefile中运行make子目录。可以做到这一点如以下：

```
subsystem:
        cd subdir && $(MAKE)

or, equivalently

subsystem:
        $(MAKE) -C subdir
```

可以编写递归复制这个例子只是通过make命令，但有很多事情，了解他们是如何和为什么工作的，以及如何子涉及到顶层make。

通信变量到子make

顶层make变量的值可以被传递到子通过环境，通过显式请求。这些变数定义子作为默认值，但不会覆盖子的makefile使用makefile中所指定的，除非使用`-e'开关

向下传递，或导出，一个变量，变量和其值的环境中运行每个命令添加。子make反过来，make使用环境变量值来初始化它的表格

特殊变量SHELL和MAKEFLAGS总是导出（除非取消导出）。 MAKEFILES导出，如果把它设置到任何东西。

如果想导出特定变量的一个子制造，使用导出指令，像这样：

```
export variable ...
```

如果想阻止一个变量被导出的，使用撤消导出的指令，像这样：

```
unexport variable ...
```

MAKEFILES 变量

MAKEFILES如果环境变量的定义，make额外的makefile 名称列表（由空格分隔）之前被读取别人认为其值。这很像include指令：不同的目录中查找这些文件。

makefile的主要用途是MAKE递归调用之间的通信

头文件包含在不同的目录

如果已经把你的头文件在不同的目录，在不同的目录中运行make，那么它需要告诉头文件的路径。这是可以做到的makefile中使用-I选项。假设该functions.h文件可在/home/yiibai/header头和其他文件/home/yiibai/src/然后进行文件将被写入如下。

```
INCLUDES = -I "/home/yiibai/header"
CC = gcc
LIBS =  -lm
CFLAGS = -g -Wall
OBJ =  main.o factorial.o hello.o

hello: ${OBJ}
   ${CC} ${CFLAGS} ${INCLUDES} -o $@ ${OBJS} ${LIBS}
.cpp.o:
   ${CC} ${CFLAGS} ${INCLUDES} -c $&lt;
```

追加更多的文本变量

通常，它用于添加更多的文字，已定义的变量的值。make 这行包含'+ =', 像这样：

```
objects += another.o
```

这需要值的变量对象，并添加文字`another.o'（前面由一个单一的空间）。因此：

```
objects = main.o hello.o factorial.o
objects += another.o
```

设置`文件main.o hello.o factorial.o another.o'的对象。

使用'+ ='是类似于：

```
objects = main.o hello.o factorial.o
objects := $(objects) another.o
```

Makefile中的续行

如果不喜欢太大的行，在Makefile中，然后你可以打破线使用反斜杠“\”，如下图所示

```
OBJ =  main.o factorial.o \
    hello.o

is equivalent to

OBJ =  main.o factorial.o hello.o
```

从命令提示符下运行的Makefile

如果已经准备好请示的Makefile的名称为“Makefile”文件，然后简单地写在命令提示符下，它将运行Makefile文件。但是，如果有任何其他的名字的Makefile，然后使用以下命令

```
make -f your-makefile-name
```

makefile 例子 - Makefile

这是一个例子编译hello程序Makefile。此程序包含三个文件main.cpp，factorial.cpp，hello.cpp。

```
# Define required macros here
SHELL = /bin/sh

OBJS =  main.o factorial.o hello.o
CFLAG = -Wall -g
CC = gcc
INCLUDE =
LIBS = -lm

hello:${OBJ}
    ${CC} ${CFLAGS} ${INCLUDES} -o $@ ${OBJS} ${LIBS}

clean:
    -rm -f *.o core *.core

.cpp.o:
    ${CC} ${CFLAGS} ${INCLUDES} -c $&lt;
```

现在可以建立hello 程序使用“make”打招呼。如果发出命令“make clean”，则它会删除所有的对象可在当前目录中的文件和核心文件。

W3School 正则表达式教程

来源：正则表达式教程

整理：飞龙

正则表达式 - 简介

除非您以前使用过正则表达式，否则您可能不熟悉此术语。但是，毫无疑问，您已经使用过不涉及脚本的某些正则表达式概念。

例如，您很可能使用 ? 和 通配符来查找硬盘上的文件。通配符匹配文件名中的单个字符，而通配符匹配零个或多个字符。像 data?.dat 这样的模式将查找下列文件：

```
data1.dat
data2.dat
datax.dat
dataN.dat
```

使用 字符代替 ? 字符扩大了找到的文件的数量。*data*.dat 匹配下列所有文件：

```
data.dat
data1.dat
data2.dat
data12.dat
datax.dat
dataXYZ.dat
```

尽管这种搜索方法很有用，但它还是有限的。通过理解 * 通配符的工作原理，引入了正则表达式所依赖的概念，但正则表达式功能更强大，而且更加灵活。

正则表达式的使用，可以通过简单的办法来实现强大的功能。下面先给出一个简单的示例：

```
 ^.+@.+\\..+$
```

继续阅读本教程将让您也可以自由应用这样的代码。

为什么使用正则表达式？

典型的搜索和替换操作要求您提供与预期的搜索结果匹配的确切文本。虽然这种技术对于对静态文本执行简单搜索和替换任务可能已经足够了，但它缺乏灵活性，若采用这种方法搜索动态文本，即使不是不可能，至少也会变得很困难。

通过使用正则表达式，可以：

- 测试字符串内的模式。
  例如，可以测试输入字符串，以查看字符串内是否出现电话号码模式或信用卡号码模式。这称为数据验证。
- 替换文本。
  可以使用正则表达式来识别文档中的特定文本，完全删除该文本或者用其他文本替换

它。
- 基于模式匹配从字符串中提取子字符串。
  可以查找文档内或输入域内特定的文本。

例如，您可能需要搜索整个网站，删除过时的材料，以及替换某些 HTML 格式标记。在这种情况下，可以使用正则表达式来确定在每个文件中是否出现该材料或该 HTML 格式标记。此过程将受影响的文件列表缩小到包含需要删除或更改的材料的那些文件。然后可以使用正则表达式来删除过时的材料。最后，可以使用正则表达式来搜索和替换标记。

发展历史

正则表达式的"祖先"可以一直上溯至对人类神经系统如何工作的早期研究。Warren McCulloch 和 Walter Pitts 这两位神经生理学家研究出一种数学方式来描述这些神经网络。

1956 年, 一位叫 Stephen Kleene 的数学家在 McCulloch 和 Pitts 早期工作的基础上，发表了一篇标题为"神经网事件的表示法"的论文，引入了正则表达式的概念。正则表达式就是用来描述他称为"正则集的代数"的表达式，因此采用"正则表达式"这个术语。

随后，发现可以将这一工作应用于使用 Ken Thompson 的计算搜索算法的一些早期研究，Ken Thompson 是 Unix 的主要发明人。正则表达式的第一个实用应用程序就是 Unix 中的 qed 编辑器。

如他们所说，剩下的就是众所周知的历史了。从那时起直至现在正则表达式都是基于文本的编辑器和搜索工具中的一个重要部分。

应用领域

目前，正则表达式已经在很多软件中得到广泛的应用，包括 *nix（Linux, Unix等）、HP 等操作系统，PHP、C#、Java 等开发环境，以及很多的应用软件中，都可以看到正则表达式的影子。

C# 正则表达式

在我们的 C# 教程中，C# 正则表达式 这一章节专门介绍了有关 C# 正则表达式的知识。

Java 正则表达式

在我们的 Java 教程中，Java 正则表达式 这一章节专门介绍了有关 Java 正则表达式的知识。

JavaScript 正则表达式

在我们的 JavaScript 教程中，JavaScript RegExp 对象 这一章节专门介绍了有关 JavaScript 正则表达式的知识，同时我们还提供了完整的 JavaScript RegExp 对象参考手册。

Python 正则表达式

在我们的 Python 基础教程中，Python 正则表达式 这一章节专门介绍了有关 Python 正则表达式的知识。

Ruby 正则表达式

在我们的 Ruby 教程中，Ruby 正则表达式 这一章节专门介绍了有关 Ruby 正则表达式的知识。

正则表达式 - 语法

正则表达式(regular expression)描述了一种字符串匹配的模式，可以用来检查一个串是否含有某种子串、将匹配的子串做替换或者从某个串中取出符合某个条件的子串等。

- 列目录时， dir *.txt*或*ls* .txt中的*.txt*就不是一个正则表达式,因为这里与正则式的*的含义是不同的。
- 构造正则表达式的方法和创建数学表达式的方法一样。也就是用多种元字符与运算符可以将小的表达式结合在一起来创建更大的表达式。正则表达式的组件可以是单个的字符、字符集合、字符范围、字符间的选择或者所有这些组件的任意组合。

正则表达式是由普通字符（例如字符 a 到 z）以及特殊字符（称为"元字符"）组成的文字模式。模式描述在搜索文本时要匹配的一个或多个字符串。正则表达式作为一个模板，将某个字符模式与所搜索的字符串进行匹配。

普通字符

普通字符包括没有显式指定为元字符的所有可打印和不可打印字符。这包括所有大写和小写字母、所有数字、所有标点符号和一些其他符号。

非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

字符	描述
\cx	匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

特殊字符

所谓特殊字符，就是一些有特殊含义的字符，如上面说的".*txt*"中的，简单的说就是表示任何字符串的意思。如果要查找文件名中有的文件，则需要对进行转义，即在其前加一个\。ls *.txt。

许多元字符要求在试图匹配它们时特别对待。若要匹配这些特殊字符，必须首先使字符"转义"，即，将反斜杠字符 () 放在它们前面。下表列出了正则表达式中的特殊字符：

特别字符	描述
$	匹配输入字符串的结尾位置。如果设置了 RegExp 对象的 Multiline 属性，则 $ 也匹配 '\n' 或 '\r'。要匹配 $ 字符本身，请使用 $。
( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 (和)。
*	匹配前面的子表达式零次或多次。要匹配 字符，请使用 \。
+	匹配前面的子表达式一次或多次。要匹配 + 字符，请使用 +。
.	匹配除换行符 \n之外的任何单字符。要匹配 .，请使用 \。
[	标记一个中括号表达式的开始。要匹配 [，请使用 [。
?	匹配前面的子表达式零次或一次，或指明一个非贪婪限定符。要匹配 ? 字符，请使用 \?。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符。序列 '\' 匹配 "\"，而 '(' 则匹配 "("。
^	匹配输入字符串的开始位置，除非在方括号表达式中使用，此时它表示不接受该字符集合。要匹配 ^ 字符本身，请使用 \^。
{	标记限定符表达式的开始。要匹配 {，请使用 {。
|	指明两项之间的一个选择。要匹配 |，请使用 \ |。

限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有*或+或?或{n}或{n,}或{n,m}共6种。

正则表达式的限定符有：

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo 能匹配 *"z"* 以及 *"zoo"*。 等价于 {0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

由于章节编号在大的输入文档中会很可能超过九，所以您需要一种方式来处理两位或三位章节编号。限定符给您这种能力。下面的正则表达式匹配编号为任何位数的章节标题：

```
/Chapter [1-9][0-9]*/
```

请注意，限定符出现在范围表达式之后。因此，它应用于整个范围表达式，在本例中，只指定从 0 到 9 的数字（包括 0 和 9）。

这里不使用 + 限定符，因为在第二个位置或后面的位置不一定需要有一个数字。也不使用 ? 字符，因为它将章节编号限制到只有两位数。您需要至少匹配 Chapter 和空格字符后面的一个数字。

如果您知道章节编号被限制为只有 99 章，可以使用下面的表达式来至少指定一位但至多两位数字。

```
/Chapter [0-9]{1,2}/
```

上面的表达式的缺点是，大于 99 的章节编号仍只匹配开头两位数字。另一个缺点是 Chapter 0 也将匹配。只匹配两位数字的更好的表达式如下：

```
/Chapter [1-9][0-9]?/
```

或

```
/Chapter [1-9][0-9]{0,1}/
```

*、+和?限定符都是贪婪的，因为它们会尽可能多的匹配文字，只有在它们的后面加上一个?就可以实现非贪婪或最小匹配。

例如，您可能搜索 HTML 文档，以查找括在 H1 标记内的章节标题。该文本在您的文档中如下：

```
<H1>Chapter 1 – Introduction to Regular Expressions</H1>
```

下面的表达式匹配从开始小于符号 (<) 到关闭 H1 标记的大于符号 (>) 之间的所有内容。

```
/<.*>/
```

如果您只需要匹配开始 H1 标记，下面的"非贪心"表达式只匹配 <H1>。

```
/<.*?>/
```

通过在 *、+ 或 ? 限定符之后放置 ?，该表达式从"贪心"表达式转换为"非贪心"表达式或者最小匹配。

定位符

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

定位符用来描述字符串或单词的边界，^和$分别指字符串的开始与结束，\b描述单词的前或后边界，\B表示非单词边界。

正则表达式的限定符有：

字符	描述
^	匹配输入字符串开始的位置。如果设置了 RegExp 对象的 Multiline 属性，^ 还会与 \n 或 \r 之后的位置匹配。
$	匹配输入字符串结尾的位置。如果设置了 RegExp 对象的 Multiline 属性，$ 还会与 \n 或 \r 之前的位置匹配。
\b	匹配一个字边界，即字与空格间的位置。
\B	非字边界匹配。

注意：不能将限定符与定位点一起使用。由于在紧靠换行或者字边界的前面或后面不能有一个以上位置，因此不允许诸如 ^* 之类的表达式。

若要匹配一行文本开始处的文本，请在正则表达式的开始使用 ^ 字符。不要将 ^ 的这种用法与中括号表达式内的用法混淆。

若要匹配一行文本的结束处的文本，请在正则表达式的结束处使用 $ 字符。

若要在搜索章节标题时使用定位点，下面的正则表达式匹配一个章节标题，该标题只包含两个尾随数字，并且出现在行首：

```
/^Chapter [1-9][0-9]{0,1}/
```

真正的章节标题不仅出现行的开始处，而且它还是该行中仅有的文本。它即出现在行首又出现在同一行的结尾。下面的表达式能确保指定的匹配只匹配章节而不匹配交叉引用。通过创建只匹配一行文本的开始和结尾的正则表达式，就可做到这一点。

```
/^Chapter [1-9][0-9]{0,1}$/
```

匹配字边界稍有不同，但向正则表达式添加了很重要的能力。字边界是单词和空格之间的位置。非字边界是任何其他位置。下面的表达式匹配单词 Chapter 的开头三个字符，因为这三个字符出现字边界后面：

```
/\bCha/
```

\b 字符的位置是非常重要的。如果它位于要匹配的字符串的开始，它在单词的开始处查找匹配项。如果它位于字符串的结尾，它在单词的结尾处查找匹配项。例如，下面的表达式匹配单词 Chapter 中的字符串 ter，因为它出现在字边界的前面：

```
/ter\b/
```

下面的表达式匹配 Chapter 中的字符串 apt，但不匹配 aptitude 中的字符串 apt：

```
/\Bapt/
```

字符串 apt 出现在单词 Chapter 中的非字边界处，但出现在单词 aptitude 中的字边界处。对于 \B 非字边界运算符，位置并不重要，因为匹配不关心究竟是单词的开头还是结尾。

选择

用圆括号将所有选择项括起来，相邻的选择项之间用|分隔。但用圆括号会有一个副作用，是相关的匹配会被缓存，此时可用?:放在第一个选项前来消除这种副作用。

其中?:是非捕获元之一，还有两个非捕获元是?=和?!，这两个还有更多的含义，前者为正向预查，在任何开始匹配圆括号内的正则表达式模式的位置来匹配搜索字符串，后者为负向预查，在任何开始不匹配该正则表达式模式的位置来匹配搜索字符串。

反向引用

对一个正则表达式模式或部分模式两边添加圆括号将导致相关匹配存储到一个临时缓冲区中，所捕获的每个子匹配都按照在正则表达式模式中从左到右出现的顺序存储。缓冲区编号从 1 开始，最多可存储 99 个捕获的子表达式。每个缓冲区都可以使用 '\n' 访问，其中 n 为一个标识特定缓冲区的一位或两位十进制数。

可以使用非捕获元字符 '?:'、'?=' 或 '?!' 来重写捕获，忽略对相关匹配的保存。

反向引用的最简单的、最有用的应用之一，是提供查找文本中两个相同的相邻单词的匹配项的能力。以下面的句子为例：

```
Is is the cost of of gasoline going up up?
```

上面的句子很显然有多个重复的单词。如果能设计一种方法定位该句子，而不必查找每个单词的重复出现，那该有多好。下面的正则表达式使用单个子表达式来实现这一点：

```
/\b([a-z]+) \1\b/gi
```

捕获的表达式，正如 [a-z]+ 指定的，包括一个或多个字母。正则表达式的第二部分是对以前捕获的子匹配项的引用，即，单词的第二个匹配项正好由括号表达式匹配。\1 指定第一个子匹配项。字边界元字符确保只检测整个单词。否则，诸如"is issued"或"this is"之类的词组将不能正确地被此表达式识别。

正则表达式后面的全局标记 (g) 指示，将该表达式应用到输入字符串中能够查找到的尽可能多的匹配。表达式的结尾处的不区分大小写 (i) 标记指定不区分大小写。多行标记指定换行符的两边可能出现潜在的匹配。

反向引用还可以将通用资源指示符 (URI) 分解为其组件。假定您想将下面的 URI 分解为协议（ftp、http 等等）、域地址和页/路径：

```
http://www.w3cschool.cc:80/html/html-tutorial.html
```

下面的正则表达式提供该功能：

```
/(\w+):\/\/([^/:]+)(:\d*)?([^# ]*)/
```

第一个括号子表达式捕获 Web 地址的协议部分。该子表达式匹配在冒号和两个正斜杠前面的任何单词。第二个括号子表达式捕获地址的域地址部分。子表达式匹配 / 或 : 之外的一个或多个字符。第三个括号子表达式捕获端口号（如果指定了的话）。该子表达式匹配冒号后面的零个或多个数字。只能重复一次该子表达式。最后，第四个括号子表达式捕获 Web 地址指定的路径和/或页信息。该子表达式能匹配不包括 # 或空格字符的任何字符序列。

将正则表达式应用到上面的 URI，各子匹配项包含下面的内容：

- 第一个括号子表达式包含"http"
- 第二个括号子表达式包含"www.w3cschool.cc"
- 第三个括号子表达式包含":80"
- 第四个括号子表达式包含"/html/html-tutorial.html"

正则表达式 - 元字符

下表包含了元字符的完整列表以及它们在正则表达式上下文中的行为：

字符	描述
\	将下一个字符标记为一个特殊字符、或一个原义字符、或一个 向后引用、或一个八进制转义符。例如，'n' 匹配字符 "n"。'\n' 匹配一个换行符。序列 '\\' 匹配 "\" 而 "\(" 则匹配 "("。
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 '\n' 或 '\r' 之后的位置。
$	匹配输入字符串的结束位置。如果设置了RegExp 对象的 Multiline 属性，$ 也匹配 '\n' 或 '\r' 之前的位置。
*	匹配前面的子表达式零次或多次。例如，zo 能匹配 *"z"* 以及 *"zoo"*。 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。
?	当该字符紧跟在任何一个其他限制符 (*, +, ?, {n}, {n,}, {n,m}) 后面时， 匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如， 对于字符串 "oooo"，'o+?' 将匹配单个 "o"，而 'o+' 将匹配所有 'o'。
.	匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 的模式。
(pattern)	匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在VBScript 中使用 SubMatches 集合，在JScript 中则使用 $0…$9 属性。要匹配圆括号字符，请使用 '\(' 或 '\)'。
(?:pattern)	匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用 "或" 字符 (|) 来组合一个模式的各个部分是很有用。例如， 'industr(?:y|ies) 就是一个比 'industry|industries' 更简略的表达式。
	正向预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个

(?=pattern)	(?=95|98|NT|2000)' 能匹配 "Windows 2000" 中的 "Windows"， 但不能匹配 "Windows 3.1" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
(?!pattern)	负向预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如'Windows (?!95|98|NT|2000)' 能匹配 "Windows 3.1" 中的 "Windows"，但不能匹配 "Windows 2000" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
x|y	匹配 x 或 y。例如，'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。
[xyz]	字符集合。匹配所包含的任意一个字符。例如， '[abc]' 可以匹配 "plain" 中的 'a'。
[^xyz]	负值字符集合。匹配未包含的任意字符。例如， '[^abc]' 可以匹配 "plain" 中的'p'。
[a-z]	字符范围。匹配指定范围内的任意字符。例如，'[a-z]' 可以匹配 'a' 到 'z' 范围内的任意小写字母字符。
[^a-z]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，'[^a-z]' 可以匹配任何不在 'a' 到 'z' 范围内的任意字符。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。
\B	匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。
\cx	匹配由 x 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。
	匹配 n，其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数

\xn	字长。例如，'\x41' 匹配 "A"。'\x041' 则等价于 '\x04' & "1"。正则表达式中可以使用 ASCII 编码。
\num	匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。例如，'(.)\1' 匹配两个连续的相同字符。
\n	标识一个八进制转义值或一个向后引用。如果 \n 之前至少 n 个获取的子表达式，则 n 为向后引用。否则，如果 n 为八进制数字 (0-7)，则 n 为一个八进制转义值。
\nm	标识一个八进制转义值或一个向后引用。如果 \nm 之前至少有 nm 个获得子表达式，则 nm 为向后引用。如果 \nm 之前至少有 n 个获取，则 n 为一个后跟文字 m 的向后引用。如果前面的条件都不满足，若 n 和 m 均为八进制数字 (0-7)，则 \nm 将匹配八进制转义值 nm。
\nml	如果 n 为八进制数字 (0-3)，且 m 和 l 均为八进制数字 (0-7)，则匹配八进制转义值 nml。
\un	匹配 n，其中 n 是一个用四个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号 (?)。

正则表达式 - 运算符优先级

正则表达式从左到右进行计算，并遵循优先级顺序，这与算术表达式非常类似。

相同优先级的从左到右进行运算，不同优先级的运算先高后低。下表从最高到最低说明了各种正则表达式运算符的优先级顺序：

运算符	描述
\	转义符
(), (?:), (?=), []	圆括号和方括号
*, +, ?, {n}, {n,}, {n,m}	限定符
^, $, \任何元字符、任何字符	定位点和序列（即：位置和顺序）
|	替换，"或"操作 字符具有高于替换运算符的优先级，使得"m|food"匹配"m"或"food"。若要匹配"mood"或"food"，请使用括号创建子表达式，从而产生"(m|f)ood"。

正则表达式 - 匹配规则

基本模式匹配

一切从最基本的开始。模式，是正规表达式最基本的元素，它们是一组描述字符串特征的字符。模式可以很简单，由普通的字符串组成，也可以非常复杂，往往用特殊的字符表示一个范围内的字符、重复出现，或表示上下文。例如：

```
^once
```

这个模式包含一个特殊的字符^，表示该模式只匹配那些以once开头的字符串。例如该模式与字符串"once upon a time"匹配，与"There once was a man from NewYork"不匹配。正如如^符号表示开头一样，$符号用来匹配那些以给定模式结尾的字符串。

```
bucket$
```

这个模式与"Who kept all of this cash in a bucket"匹配，与"buckets"不匹配。字符^和$同时使用时，表示精确匹配（字符串与模式一样）。例如：

```
^bucket$
```

只匹配字符串"bucket"。如果一个模式不包括^和$，那么它与任何包含该模式的字符串匹配。例如：模式

```
once
```

与字符串

```
There once was a man from NewYork
Who kept all of his cash in a bucket.
```

是匹配的。

在该模式中的字母(o-n-c-e)是字面的字符，也就是说，他们表示该字母本身，数字也是一样的。其他一些稍微复杂的字符，如标点符号和白字符（空格、制表符等），要用到转义序列。所有的转义序列都用反斜杠()打头。制表符的转义序列是：\t。所以如果我们要检测一个字符串是否以制表符开头，可以用这个模式：

```
^\t
```

类似的，用\n表示"新行"，\r表示回车。其他的特殊符号，可以用在前面加上反斜杠，如反斜杠本身用\表示，句号.用.表示，以此类推。

字符簇

在INTERNET的程序中，正规表达式通常用来验证用户的输入。当用户提交一个FORM以后，要判断输入的电话号码、地址、EMAIL地址、信用卡号码等是否有效，用普通的基于字面的字符是不够的。

所以要用一种更自由的描述我们要的模式的办法，它就是字符簇。要建立一个表示所有元音字符的字符簇，就把所有的元音字符放在一个方括号里：

```
[AaEeIiOoUu]
```

这个模式与任何元音字符匹配，但只能表示一个字符。用连字号可以表示一个字符的范围，如：

```
[a-z] //匹配所有的小写字母
[A-Z] //匹配所有的大写字母
[a-zA-Z] //匹配所有的字母
[0-9] //匹配所有的数字
[0-9\.\-] //匹配所有的数字，句号和减号
[ \f\r\t\n] //匹配所有的白字符
```

同样的，这些也只表示一个字符，这是一个非常重要的。如果要匹配一个由一个小写字母和一位数字组成的字符串，比如"z2"、"t6"或"g7"，但不是"ab2"、"r2d3" 或"b52"的话，用这个模式：

```
^[a-z][0-9]$
```

尽管[a-z]代表26个字母的范围，但在这里它只能与第一个字符是小写字母的字符串匹配。

前面曾经提到^表示字符串的开头，但它还有另外一个含义。当在一组方括号里使用^是，它表示"非"或"排除"的意思，常常用来剔除某个字符。还用前面的例子，我们要求第一个字符不能是数字：

```
^[^0-9][0-9]$
```

这个模式与"&5"、"g7"及"-2"是匹配的，但与"12"、"66"是不匹配的。下面是几个排除特定字符的例子：

```
[^a-z] //除了小写字母以外的所有字符
[^\\\/\^] //除了(\)(/)(^)之外的所有字符
[^\"\'] //除了双引号(")和单引号(')之外的所有字符
```

特殊字符"." (点，句号)在正规表达式中用来表示除了"新行"之外的所有字符。所以模式"^.5$"与任何两个字符的、以数字5结尾和以其他非"新行"字符开头的字符串匹配。模式"."可以匹配任何字符串，除了空串和只包括一个"新行"的字符串。

PHP的正规表达式有一些内置的通用字符簇，列表如下：

字符簇	描述
[[:alpha:]]	任何字母
[[:digit:]]	任何数字
[[:alnum:]]	任何字母和数字
[[:space:]]	任何空白字符
[[:upper:]]	任何大写字母
[[:lower:]]	任何小写字母
[[:punct:]]	任何标点符号
[[:xdigit:]]	任何16进制的数字，相当于[0-9a-fA-F]

确定重复出现

到现在为止，你已经知道如何去匹配一个字母或数字，但更多的情况下，可能要匹配一个单词或一组数字。一个单词有若干个字母组成，一组数字有若干个单数组成。跟在字符或字符簇后面的花括号({})用来确定前面的内容的重复出现的次数。

字符簇	描述
^[a-zA-Z_]$	所有的字母和下划线
^[[:alpha:]]{3}$	所有的3个字母的单词
^a$	字母a
^a{4}$	aaaa
^a{2,4}$	aa,aaa或aaaa
^a{1,3}$	a,aa或aaa
^a{2,}$	包含多于两个a的字符串
^a{2,}	如：aardvark和aaab，但apple不行
a{2,}	如：baad和aaa，但Nantucket不行
\t{2}	两个制表符
.{2}	所有的两个字符

这些例子描述了花括号的三种不同的用法。一个数字，{x}的意思是"前面的字符或字符簇只出现x次"；一个数字加逗号，{x,}的意思是"前面的内容出现x或更多的次数"；两个用逗号分隔的数字，{x,y}表示"前面的内容至少出现x次，但不超过y次"。我们可以把模式扩展到更多的单词或数字：

```
^[a-zA-Z0-9_]{1,}$ //所有包含一个以上的字母、数字或下划线的字符串
^[0-9]{1,}$ //所有的正数
^\-{0,1}[0-9]{1,}$ //所有的整数
^\-{0,1}[0-9]{0,}\.{0,1}[0-9]{0,}$ //所有的小数
```

最后一个例子不太好理解，是吗？这么看吧：与所有以一个可选的负号(-{0,1})开头(^)、跟着0个或更多的数字([0-9]{0,})、和一个可选的小数点(.{0,1})再跟上0个或多个数字([0-9]{0,})，并且没有其他任何东西($)。下面你将知道能够使用的更为简单的方法。

特殊字符"?"与{0,1}是相等的，它们都代表着："0个或1个前面的内容"或"前面的内容是可选的"。所以刚才的例子可以简化为：

```
^\-?[0-9]{0,}\.?[0-9]{0,}$
```

特殊字符"*"与{0,}是相等的，它们都代表着"0个或多个前面的内容"。最后，字符"+"与 {1,}是相等的，表示"1个或多个前面的内容"，所以上面的4个例子可以写成：

```
^[a-zA-Z0-9_]+$ //所有包含一个以上的字母、数字或下划线的字符串
^[0-9]+$ //所有的正数
^\-?[0-9]+$ //所有的整数
^\-?[0-9]*\.?[0-9]*$ //所有的小数
```

当然这并不能从技术上降低正规表达式的复杂性，但可以使它们更容易阅读。

正则表达式 - 示例

简单表达式

正则表达式的最简单形式是在搜索字符串中匹配其本身的单个普通字符。例如，单字符模式，如 A，不论出现在搜索字符串中的何处，它总是匹配字母 A。下面是一些单字符正则表达式模式的示例：

```
/a/
/7/
/M/
```

可以将许多单字符组合起来以形成大的表达式。例如，以下正则表达式组合了单字符表达式：a、7 和 M。

```
/a7M/
```

请注意，没有串联运算符。只须在一个字符后面键入另一个字符。

字符匹配

句点 (.) 匹配字符串中的各种打印或非打印字符，只有一个字符例外。这个例外就是换行符 (\n)。下面的正则表达式匹配 aac、abc、acc、adc 等等，以及 a1c、a2c、a-c 和 a#c：

```
/a.c/
```

若要匹配包含文件名的字符串，而句点 (.) 是输入字符串的组成部分，请在正则表达式中的句点前面加反斜扛 () 字符。举例来说明，下面的正则表达式匹配 filename.ext：

```
/filename\.ext/
```

这些表达式只让您匹配"任何"单个字符。可能需要匹配列表中的特定字符组。例如，可能需要查找用数字表示的章节标题（Chapter 1、Chapter 2 等等）。

中括号表达式

若要创建匹配字符组的一个列表，请在方括号（[和]）内放置一个或更多单个字符。当字符括在中括号内时，该列表称为"中括号表达式"。与在任何别的位置一样，普通字符在中括号内表示其本身，即，它在输入文本中匹配一次其本身。大多数特殊字符在中括号表达式内出现时失去它们的意义。不过也有一些例外，如：

- 如果] 字符不是第一项，它结束一个列表。若要匹配列表中的] 字符，请将它放在第一位，紧跟在开始 [后面。
- \ 字符继续作为转义符。若要匹配 \ 字符，请使用 \\。

括在中括号表达式中的字符只匹配处于正则表达式中该位置的单个字符。以下正则表达式匹配 Chapter 1、Chapter 2、Chapter 3、Chapter 4 和 Chapter 5：

```
/Chapter [12345]/
```

请注意，单词 Chapter 和后面的空格的位置相对于中括号内的字符是固定的。中括号表达式指定的只是匹配紧跟在单词 Chapter 和空格后面的单个字符位置的字符集。这是第九个字符位置。

若要使用范围代替字符本身来表示匹配字符组，请使用连字符 (-) 将范围中的开始字符和结束字符分开。单个字符的字符值确定范围内的相对顺序。下面的正则表达式包含范围表达式，该范围表达式等效于上面显示的中括号中的列表。

```
/Chapter [1-5]/
```

当以这种方式指定范围时，开始值和结束值两者都包括在范围内。注意，还有一点很重要，按 Unicode 排序顺序，开始值必须在结束值的前面。

若要在中括号表达式中包括连字符，请采用下列方法之一：

- 用反斜扛将它转义：

  ```
  [\-]
  ```

- 将连字符放在中括号列表的开始或结尾。下面的表达式匹配所有小写字母和连字符：

  ```
  [-a-z]
  [a-z-]
  ```

- 创建一个范围，在该范围中，开始字符值小于连字符，而结束字符值等于或大于连字符。下面的两个正则表达式都满足这一要求：

  ```
  [!--]
  [!-~]
  ```

若要查找不在列表或范围内的所有字符，请将插入符号 (^) 放在列表的开头。如果插入字符出现在列表中的其他任何位置，则它匹配其本身。下面的正则表达式匹配编号大于 5 的章节标题：

```
/Chapter [^12345]/
```

在上面的示例中，表达式在第九个位置匹配 1、2、3、4 或 5 之外的任何数字字符。这样，例如，Chapter 7 就是一个匹配项，Chapter 9 也是一个匹配项。

上面的表达式可以使用连字符 (-) 来表示：

```
/Chapter [^1-5]/
```

中括号表达式的典型用途是指定任何大写或小写字母或任何数字的匹配。下面的表达式指定这样的匹配：

```
/[A-Za-z0-9]/
```

替换和分组

替换使用 | 字符来允许在两个或多个替换选项之间进行选择。例如，可以扩展章节标题正则表达式，以返回比章标题范围更广的匹配项。但是，这并不象您可能认为的那样简单。替换匹配 | 字符任一侧最大的表达式。

您可能认为，下面的表达式匹配出现在行首和行尾、后面跟一个或两个数字的 Chapter 或 Section：

```
/^Chapter|Section [1-9][0-9]{0,1}$/
```

很遗憾，上面的正则表达式要么匹配行首的单词 Chapter，要么匹配行尾的单词 Section 及跟在其后的任何数字。如果输入字符串是 Chapter 22，那么上面的表达式只匹配单词 Chapter。如果输入字符串是 Section 22，那么该表达式匹配 Section 22。

若要使正则表达式更易于控制，可以使用括号来限制替换的范围，即，确保它只应用于两个单词 Chapter 和 Section。但是，括号也用于创建子表达式，并可能捕获它们以供以后使用，这一点在有关反向引用的那一节讲述。通过在上面的正则表达式的适当位置添加括号，就可以使该正则表达式匹配 Chapter 1 或 Section 3。

下面的正则表达式使用括号来组合 Chapter 和 Section，以便表达式正确地起作用：

```
/^(Chapter|Section) [1-9][0-9]{0,1}$/
```

尽管这些表达式正常工作，但 Chapter|Section 周围的括号还将捕获两个匹配字中的任一个供以后使用。由于在上面的表达式中只有一组括号，因此，只有一个被捕获的"子匹配项"。

在上面的示例中，您只需要使用括号来组合单词 Chapter 和 Section 之间的选择。若要防止匹配被保存以备将来使用，请在括号内正则表达式模式之前放置 ?:。下面的修改提供相同的能力而不保存子匹配项：

```
/^(?:Chapter|Section) [1-9][0-9]{0,1}$/
```

除 ?: 元字符外，两个其他非捕获元字符创建被称为"预测先行"匹配的某些内容。正向预测先行使用 ?= 指定，它匹配处于括号中匹配正则表达式模式的起始点的搜索字符串。反向预测先行使用 ?! 指定，它匹配处于与正则表达式模式不匹配的字符串的起始点的搜索字符串。

例如，假设您有一个文档，该文档包含指向 Windows 3.1、Windows 95、Windows 98 和 Windows NT 的引用。再进一步假设，您需要更新该文档，将指向 Windows 95、Windows 98 和 Windows NT 的所有引用更改为 Windows 2000。下面的正则表达式（这是一个正向预测先行的示例）匹配 Windows 95、Windows 98 和 Windows NT：

```
/Windows(?=95 |98 |NT )/
```

找到一处匹配后，紧接着就在匹配的文本（不包括预测先行中的字符）之后搜索下一处匹配。例如，如果上面的表达式匹配 Windows 98，将在 Windows 之后而不是在 98 之后继续搜索。

其他示例

下面列出一些正则表达式示例：

<table>
<tr><th>正则表达式</th><th>描述</th></tr>
<tr><td>/\b([a-z]+) \1\b/gi</td><td>一个单词连续出现的位置。</td></tr>
<tr><td>/(\w+):\/\/([^/:]+)(:\d)?([^#])/</td><td>将一个URL解析为协议、域、端口及相对路径。</td></tr>
<tr><td>/^(?:Chapter|Section) [1-9][0-9]{0,1}$/</td><td>定位章节的位置。</td></tr>
<tr><td>/[-a-z]/</td><td>A至z共26个字母再加一个-号。</td></tr>
<tr><td>/ter\b/</td><td>可匹配chapter，而不能匹配terminal。</td></tr>
<tr><td>/\Bapt/</td><td>可匹配chapter，而不能匹配aptitude。</td></tr>
<tr><td>/Windows(?=95 |98 |NT)/</td><td>可匹配Windows95或Windows98或WindowsNT，当找到一个匹配后，从Windows后面开始进行下一次的检索匹配。</td></tr>
<tr><td>/^\s*$/</td><td>匹配空行。</td></tr>
<tr><td>/\d{2}-\d{5}/</td><td>验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。</td></tr>
<tr><td>/<\s*(\S+)(\s[^>]*)?>[\s\S]*<\s*\/\1\s*>/</td><td>匹配 HTML 标记。</td></tr>
</table>

Shell 编程

Shell 教程

Shell 是一个用C语言编写的程序，它是用户使用Linux的桥梁。Shell既是一种命令语言，又是一种程序设计语言。

Shell 是指一种应用程序，这个应用程序提供了一个界面，用户通过这个界面访问操作系统内核的服务。

Ken Thompson的sh是第一种Unix Shell，Windows Explorer是一个典型的图形界面Shell。

Shell 脚本

Shell 脚本（shell script），是一种为shell编写的脚本程序。

业界所说的shell通常都是指shell脚本，但读者朋友要知道，shell和shell script是两个不同的概念。

由于习惯的原因，简洁起见，本文出现的"shell编程"都是指shell脚本编程，不是指开发shell自身。

Shell 环境

Shell 编程跟java、php编程一样，只要有一个能编写代码的文本编辑器和一个能解释执行的脚本解释器就可以了。

Linux的Shell种类众多，常见的有：

- Bourne Shell（/usr/bin/sh或/bin/sh）
- Bourne Again Shell（/bin/bash）
- C Shell（/usr/bin/csh）
- K Shell（/usr/bin/ksh）
- Shell for Root（/sbin/sh）
- ……

本教程关注的是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash在日常工作中被广泛使用。同时，Bash也是大多数Linux系统默认的Shell。

在一般情况下，人们并不区分 Bourne Shell 和 Bourne Again Shell，所以，像 **#!/bin/sh**，它同样也可以改为**#!/bin/bash**。

`#!` 告诉系统其后路径所指定的程序即是解释此脚本文件的Shell程序。

第一个shell脚本

打开文本编辑器(可以使用vi/vim命令来创建文件)，新建一个文件test.sh，扩展名为sh（sh代表shell），扩展名并不影响脚本执行，见名知意就好，如果你用php写shell 脚本，扩展名就用php好了。

输入一些代码，第一行一般是这样：

```
#!/bin/bash
echo "Hello World !"
```

" `#!` " 是一个约定的标记，它告诉系统这个脚本需要什么解释器来执行，即使用哪一种Shell。

echo命令用于向窗口输出文本。

运行Shell脚本有两种方法：

1、作为可执行程序

将上面的代码保存为test.sh，并cd到相应目录：

```
chmod +x ./test.sh  #使脚本具有执行权限
./test.sh  #执行脚本
```

注意，一定要写成./test.sh，而不是test.sh，运行其它二进制的程序也一样，直接写test.sh，linux系统会去PATH里寻找有没有叫test.sh的，而只有/bin, /sbin, /usr/bin，/usr/sbin等在PATH里，你的当前目录通常不在PATH里，所以写成test.sh是会找不到命令的，要用./test.sh告诉系统说，就在当前目录找。

2、作为解释器参数

这种运行方式是，直接运行解释器，其参数就是shell脚本的文件名，如：

```
/bin/sh test.sh
/bin/php test.php
```

这种方式运行的脚本，不需要在第一行指定解释器信息，写了也没用。

Shell 变量

定义变量时，变量名不加美元符号（$，PHP语言中变量需要），如：

```
your_name="w3cschool.cc"
```

注意，变量名和等号之间不能有空格，这可能和你熟悉的所有编程语言都不一样。同时，变量名的命名须遵循如下规则：

- 首个字符必须为字母（a-z，A-Z）。
- 中间不能有空格，可以使用下划线（_）。
- 不能使用标点符号。
- 不能使用bash里的关键字（可用help命令查看保留关键字）。

除了显式地直接赋值，还可以用语句给变量赋值，如：

```
for file in `ls /etc`
```

以上语句将 /etc 下目录的文件名循环出来。

使用变量

使用一个定义过的变量，只要在变量名前面加美元符号即可，如：

```
your_name="qinjx"
echo $your_name
echo ${your_name}
```

变量名外面的花括号是可选的，加不加都行，加花括号是为了帮助解释器识别变量的边界，比如下面这种情况：

```
for skill in Ada Coffe Action Java do
    echo "I am good at ${skill}Script"
done
```

如果不给skill变量加花括号，写成echo "I am good at $skillScript"，解释器就会把$skillScript当成一个变量（其值为空），代码执行结果就不是我们期望的样子了。

推荐给所有变量加上花括号，这是个好的编程习惯。

已定义的变量，可以被重新定义，如：

```
your_name="tom"
echo $your_name
your_name="alibaba"
echo $your_name
```

这样写是合法的，但注意，第二次赋值的时候不能写$your_name="alibaba"，使用变量的时候才加美元符（$）。

Shell 字符串

字符串是shell编程中最常用最有用的数据类型（除了数字和字符串，也没啥其它类型好用了），字符串可以用单引号，也可以用双引号，也可以不用引号。单双引号的区别跟PHP类似。

单引号

```
str='this is a string'
```

单引号字符串的限制：

- 单引号里的任何字符都会原样输出，单引号字符串中的变量是无效的；
- 单引号字串中不能出现单引号（对单引号使用转义符后也不行）。

双引号

```
your_name='qinjx'
str="Hello, I know your are \"$your_name\"! \n"
```

双引号的优点：

- 双引号里可以有变量
- 双引号里可以出现转义字符

拼接字符串

```
your_name="qinjx"
greeting="hello, "$your_name" !"
greeting_1="hello, ${your_name} !"
echo $greeting $greeting_1
```

获取字符串长度

```
string="abcd"
echo ${#string} #输出 4
```

提取子字符串

```
string="alibaba is a great company"
echo ${string:1:4} #输出liba
```

查找子字符串

```
string="alibaba is a great company"
echo `expr index "$string" is`
```

Shell 数组

bash支持一维数组（不支持多维数组），并且没有限定数组的大小。

类似与C语言，数组元素的下标由0开始编号。获取数组中的元素要利用下标，下标可以是整数或算术表达式，其值应大于或等于0。

定义数组

在Shell中，用括号来表示数组，数组元素用"空格"符号分割开。定义数组的一般形式为：

```
    数组名=(值1 值2 ... 值n)
```

例如：

```
array_name=(value0 value1 value2 value3)
```

或者

```
array_name=(
value0
value1
value2
value3
)
```

还可以单独定义数组的各个分量：

```
array_name[0]=value0
array_name[1]=value1
array_name[n]=valuen
```

可以不使用连续的下标，而且下标的范围没有限制。

读取数组

读取数组元素值的一般格式是：

```
    ${数组名[下标]}
```

例如：

```
valuen=${array_name[n]}
```

使用@符号可以获取数组中的所有元素，例如：

```
echo ${array_name[@]}
```

获取数组的长度

获取数组长度的方法与获取字符串长度的方法相同，例如：

```
# 取得数组元素的个数
length=${#array_name[@]}
# 或者
length=${#array_name[*]}
# 取得数组单个元素的长度
lengthn=${#array_name[n]}
```

Shell 注释

以"#"开头的行就是注释，会被解释器忽略。

sh里没有多行注释，只能每一行加一个#号。只能像这样：

```
#--------------------------------------------
# 这是一个自动打ipa的脚本，基于webfrogs的ipa-build书写：
# https://github.com/webfrogs/xcode_shell/blob/master/ipa-build
# 功能：自动为etao ios app打包，产出物为14个渠道的ipa包
# 特色：全自动打包，不需要输入任何参数
#--------------------------------------------
##### 用户配置区 开始 #####
#
#
# 项目根目录，推荐将此脚本放在项目的根目录，这里就不用改了
# 应用名，确保和Xcode里Product下的target_name.app名字一致
#
##### 用户配置区 结束  #####
```

如果在开发过程中，遇到大段的代码需要临时注释起来，过一会儿又取消注释，怎么办呢？

每一行加个#符号太费力了，可以把这一段要注释的代码用一对花括号括起来，定义成一个函数，没有地方调用这个函数，这块代码就不会执行，达到了和注释一样的效果。

Shell echo命令

Shell 的 echo 指令与 PHP 的 echo 指令类似，都是用于字符串的输出。命令格式：

```
echo string
```

您可以使用echo实现更复杂的输出格式控制。

1.显示普通字符串:

```
 echo "It is a test"
```

这里的双引号完全可以省略，以下命令与上面实例效果一致：

```
 echo It is a test
```

2.显示转义字符

```
 echo "\"It is a test\""
```

结果将是:

```
 "It is a test"
```

同样，双引号也可以省略

3.显示变量

read 命令从标准输入中读取一行,并把输入行的每个字段的值指定给 shell 变量

```
#!/bin/sh
read name
echo "$name It is a test"
```

以上代码保存为 test.sh，name 接收标准输入的变量，结果将是:

```
[root@www ~]# sh test.sh
OK                     #标准输入
OK It is a test        #输出
```

4.显示换行

```
echo -e "OK!\n" # -e 开启转义
echo "It it a test"
```

输出结果：

```
OK!

It it a test
```

5.显示不换行

```
#!/bin/sh
echo -e "OK! \c" # -e 开启转义 \c 不换行
echo "It is a test"
```

输出结果：

```
OK! It is a test
```

6.显示结果定向至文件

```
echo "It is a test" > myfile
```

7.原样输出字符串，不进行转义或取变量(用单引号)

```
echo '$name\"'
```

输出结果：

```
$name\"
```

8.显示命令执行结果

```
echo `date`
```

结果将显示当前日期

```
Thu Jul 24 10:08:46 CST 2014
```

Shell test命令

Shell中的 test 命令用于检查某个条件是否成立，它可以进行数值、字符和文件三个方面的测试。

数值测试

参数	说明
-eq	等于则为真
-ne	不等于则为真
-gt	大于则为真
-ge	大于等于则为真
-lt	小于则为真
-le	小于等于则为真

实例演示：

```
num1=100
num2=100
if test $[num1] -eq $[num2]
then
    echo 'The two numbers are equal!'
else
    echo 'The two numbers are not equal!'
fi
```

输出结果：

```
The two numbers are equal!
```

字符串测试

参数	说明
=	等于则为真
!=	不相等则为真
-z 字符串	字符串长度伪则为真
-n 字符串	字符串长度不伪则为真

实例演示：

```
num1=100
num2=100
if test num1=num2
then
    echo 'The two strings are equal!'
else
    echo 'The two strings are not equal!'
fi
```

输出结果：

```
The two strings are equal!
```

文件测试

参数	说明
-e 文件名	如果文件存在则为真
-r 文件名	如果文件存在且可读则为真
-w 文件名	如果文件存在且可写则为真
-x 文件名	如果文件存在且可执行则为真
-s 文件名	如果文件存在且至少有一个字符则为真
-d 文件名	如果文件存在且为目录则为真
-f 文件名	如果文件存在且为普通文件则为真
-c 文件名	如果文件存在且为字符型特殊文件则为真
-b 文件名	如果文件存在且为块特殊文件则为真

实例演示：

```
cd /bin
if test -e ./bash
then
    echo 'The file already exists!'
else
    echo 'The file does not exists!'
fi
```

输出结果：

```
The file already exists!
```

另外，Shell还提供了与(!)、或(-o)、非(-a)三个逻辑操作符用于将测试条件连接起来，其优先级为："!"最高，"-a"次之，"-o"最低。例如：

```
cd /bin
if test -e ./notFile -o ./bash
then
    echo 'One file exists at least!'
else
    echo 'Both dose not exists!'
fi
```

输出结果：

```
One file exists at least!
```

Shell 流程控制

和Java、PHP等语言不一样，sh的流程控制不可为空，如(以下为PHP流程控制写法)：

```
<?php
if (isset($_GET["q"])) {
    search(q);
}
else {
    //do nothing
}
```

在sh/bash里可不能这么写，如果else分支没有语句执行，就不要写这个else，就像这样

if else

if

if 语句语法格式：

```
if condition
then
    command1
    command2
    ...
    commandN
fi
```

写成一行（适用于终端命令提示符）：

```
if `ps -ef | grep ssh`;  then echo hello; fi
```

末尾的fi就是if倒过来拼写，后面还会遇到类似的。

if else

if else 语法格式：

```
if condition
then
    command1
    command2
    ...
    commandN
else
    command
fi
```

if else-if else

if else-if else 语法格式：

```
if condition1
then
    command1
elif condition2
    command2
else
    commandN
fi
```

if else语句经常与test命令结合使用，如下所示：

```
num1=$[2*3]
num2=$[1+5]
if test $[num1] -eq $[num2]
then
    echo 'The two numbers are equal!'
else
    echo 'The two numbers are not equal!'
fi
```

输出结果：

```
The two numbers are equal!
```

for 循环

与其他编程语言类似，Shell支持for循环。

for循环一般格式为：

```
for var in item1 item2 ... itemN
do
    command1
    command2
    ...
    commandN
done
```

写成一行：

```
for var in item1 item2 ... itemN; do command1; command2… done;
```

当变量值在列表里，for循环即执行一次所有命令，使用变量名获取列表中的当前取值。命令可为任何有效的shell命令和语句。in列表可以包含替换、字符串和文件名。

in列表是可选的，如果不用它，for循环使用命令行的位置参数。

例如，顺序输出当前列表中的数字：

```
for loop in 1 2 3 4 5
do
    echo "The value is: $loop"
done
```

输出结果：

```
The value is: 1
The value is: 2
The value is: 3
The value is: 4
The value is: 5
```

顺序输出字符串中的字符：

```
for str in 'This is a string'
do
    echo $str
done
```

输出结果：

```
This is a string
```

while 语句

while循环用于不断执行一系列命令，也用于从输入文件中读取数据；命令通常为测试条件。其格式为：

```
while condition
do
    command
done
```

命令执行完毕，控制返回循环顶部，从头开始直至测试条件为假。

以下是一个基本的while循环，测试条件是：如果COUNTER小于5，那么条件返回真。COUNTER从0开始，每次循环处理时，COUNTER加1。运行上述脚本，返回数字1到5，然后终止。

```
COUNTER=0
while [ $COUNTER -lt 5 ]
do
    COUNTER='expr $COUNTER+1'
    echo $COUNTER
done
```

运行脚本，输出：

```
1
2
3
4
5
```

while循环可用于读取键盘信息。下面的例子中，输入信息被设置为变量FILM，按<Ctrl-D>结束循环。

```
echo 'type <CTRL-D> to terminate'
echo -n 'enter your most liked film: ''
while read FILM
do
    echo "Yeah! great film the $FILM"
done
```

运行脚本，输出类似下面：

```
type <CTRL-D> to terminate
enter your most liked film: Sound of Music
Yeah! great film the Sound of Music
```

无限循环

无限循环语法格式：

```
while :
do
    command
done
```

或者

```
while true
do
    command
done
```

或者

```
for (( ; ; ))
```

until 循环

until循环执行一系列命令直至条件为真时停止。

until循环与while循环在处理方式上刚好相反。

一般while循环优于until循环，但在某些时候—也只是极少数情况下，until循环更加有用。

until 语法格式:

```
until condition
do
    command
done
```

条件可为任意测试条件，测试发生在循环末尾，因此循环至少执行一次—请注意这一点。

case

Shell case语句为多选择语句。可以用case语句匹配一个值与一个模式，如果匹配成功，执行相匹配的命令。case语句格式如下：

```
case 值 in
模式1)
    command1
    command2
    ...
    commandN
    ;;
模式2)
    command1
    command2
    ...
    commandN
    ;;
esac
```

case工作方式如上所示。取值后面必须为单词in，每一模式必须以右括号结束。取值可以为变量或常数。匹配发现取值符合某一模式后，其间所有命令开始执行直至 ;;。

取值将检测匹配的每一个模式。一旦模式匹配，则执行完匹配模式相应命令后不再继续其他模式。如果无一匹配模式，使用星号 * 捕获该值，再执行后面的命令。

下面的脚本提示输入1到4，与每一种模式进行匹配：

```
echo 'Input a number between 1 to 4'
echo 'Your number is:\c'
read aNum
case $aNum in
    1)  echo 'You select 1'
    ;;
    2)  echo 'You select 2'
    ;;
    3)  echo 'You select 3'
    ;;
    4)  echo 'You select 4'
    ;;
    *)  echo 'You do not select a number between 1 to 4'
    ;;
esac
```

输入不同的内容，会有不同的结果，例如：

```
Input a number between 1 to 4
Your number is:3
You select 3
```

跳出循环

在循环过程中，有时候需要在未达到循环结束条件时强制跳出循环，Shell使用两个命令来实现该功能：break和continue。

break命令

break命令允许跳出所有循环（终止执行后面的所有循环）。

下面的例子中，脚本进入死循环直至用户输入数字大于5。要跳出这个循环，返回到shell提示符下，需要使用break命令。

```
#!/bin/bash
while :
do
    echo -n "Input a number between 1 to 5: "
    read aNum
    case $aNum in
        1|2|3|4|5) echo "Your number is $aNum!"
        ;;
        *) echo "You do not select a number between 1 to 5, game is over!"
            break
        ;;
    esac
done
```

continue

```
continue命令与break命令类似，只有一点差别，它不会跳出所有循环，仅仅跳出当前循环。
```

对上面的例子进行修改：

```
#!/bin/bash
while :
do
    echo -n "Input a number between 1 to 5: "
    read aNum
    case $aNum in
        1|2|3|4|5) echo "Your number is $aNum!"
        ;;
        *) echo "You do not select a number between 1 to 5!"
            continue
            echo "Game is over!"
        ;;
    esac
done
```

运行代码发现，当输入大于5的数字时，该例中的循环不会结束，语句 **echo "Game is over!"** 永远不会被执行。

esac

case的语法和C family语言差别很大，它需要一个esac（就是case反过来）作为结束标记，每个case分支用右圆括号，用两个分号表示break。

Shell 函数

linux shell 可以用户定义函数，然后在shell脚本中可以随便调用。

shell中函数的定义格式如下：

```
[ function ] funname [()]

{

    action;

    [return int;]

}
```

说明：

- 1、可以带function fun() 定义，也可以直接fun() 定义,不带任何参数。
- 2、参数返回，可以显示加：return 返回，如果不加，将以最后一条命令运行结果，作为返回值。 return后跟数值n(0-255

下面的例子定义了一个函数并进行调用：

```
#!/bin/bash
demoFun(){
    echo "This is your first shell function!"
}
echo "Function begin..."
hello
echo "Function end!"
```

输出结果：

```
Function begin...
This is your first shell function!
Function end!
```

下面定义一个带有return语句的函数：

```
#!/bin/bash
funWithReturn(){
    echo "The function is to get the sum of two numbers..."
    echo -n "Input first number: "
    read aNum
    echo -n "Input another number: "
    read anotherNum
    echo "The two numbers are $aNum and $anotherNum !"
    return $(($aNum+$anotherNum))
}
funWithReturn
echo "The sum of two numbers is $? !"
```

输出类似下面：

```
The function is to get the sum of two numbers...
Input first number: 25
Input another number: 50
The two numbers are 25 and 50 !
The sum of two numbers is 75 !
```

函数返回值在调用该函数后通过 $? 来获得。

注意：所有函数在使用前必须定义。这意味着必须将函数放在脚本开始部分，直至shell解释器首次发现它时，才可以使用。调用函数仅使用其函数名即可。

函数参数

在Shell中，调用函数时可以向其传递参数。在函数体内部，通过 $n 的形式来获取参数的值，例如，$1表示第一个参数，$2表示第二个参数...

带参数的函数示例：

```
#!/bin/bash
funWithParam(){
    echo "The value of the first parameter is $1 !"
    echo "The value of the second parameter is $2 !"
    echo "The value of the tenth parameter is $10 !"
    echo "The value of the tenth parameter is ${10} !"
    echo "The value of the eleventh parameter is ${11} !"
    echo "The amount of the parameters is $# !"
    echo "The string of the parameters is $* !"
}
funWithParam 1 2 3 4 5 6 7 8 9 34 73
```

输出结果：

```
The value of the first parameter is 1 !
The value of the second parameter is 2 !
The value of the tenth parameter is 10 !
The value of the tenth parameter is 34 !
The value of the eleventh parameter is 73 !
The amount of the parameters is 12 !
The string of the parameters is 1 2 3 4 5 6 7 8 9 34 73 !"
```

注意，$10 不能获取第十个参数，获取第十个参数需要${10}。当n>=10时，需要使用${n}来获取参数。

另外，还有几个特殊字符用来处理参数：

参数处理	说明
$#	传递到脚本的参数个数
$*	以一个单字符串显示所有向脚本传递的参数
$	脚本运行的当前进程ID号
$!	后台运行的最后一个进程的ID号
$@	与$#相同，但是使用时加引号，并在引号中返回每个参数。
$-	显示Shell使用的当前选项，与set命令功能相同。
$?	显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。

UML教程首页 - UML

UML是一种标准语言，用于指定，可视化，构造和文档的软件系统的文物。UML是OMG在1997年1月提出了创建由对象管理组和UML1.0规范草案。本教程给出了一个比较完整的学习理解UML，可以方便学习UML入门和使用。

UML工程师的相关实用链接

(1) UML快速参考指南

一个快速为UML工程师的UML参考手册

(2) 有用的UML资源

UML网站，书籍和文章的集合。

(3) UML工具和实用程序

在这里，可以找到一个有用的UML的工具和实用程序、UML建模。

UML概述 - UML

UML是一种标准语言，用于指定，可视化，构造和文档的软件系统。

UML是OMG在1997年1月提出了创建由对象管理组织（OMG）和UML1.0规范草案。

OMG不断努力，使一个真正的行业标准。

- UML 代表 **<u>U</u>**nified **<u>M</u>**odeling **<u>L</u>**anguage.
- UML是不同于其他常见的编程语言，如C + +，Java中，COBOL等。
- UML是一种绘画语言，用来做软件蓝图。

因此，UML可以作为一个通用的可视化建模语言，可视化，指定兴建及记录软件系统。虽然UML一般用于模型的软件系统，但它并不限于在此范围内。它也可以用来建模非软件系统的处理流程，以及像在一个制造单元等

UML不是一种编程语言，但工具可用于生成各种语言的代码中使用UML图。 UML面向对象的分析和设计有直接关系。经过一段标准化UML成为OMG（对象管理组织）标准。

UML目标:

一张图片胜过千言万语，这绝对适合在讨论关于UML。远远早于UML的面向对象的概念被引入。所以在那个时候，有没有标准的方法来组织和整合面向对象的发展。在那个时间点UML进入图像。

有许多开发UML的目标，但最重要的是定义一些通用的建模语言，建模者可以使用，也需要作出简单的理解和使用。

UML图不仅也为企业用户，普通人和有兴趣的人来了解系统的开发。该系统可以是一个软件或使用非软件。因此，它必须是明确的，UML不是一种开发方法，而伴随着流程，做一个成功的系统。

在总结可以被定义为一个简单的建模机制，在当今复杂的环境中所有可能的实际系统进行建模是UML的目标。

UML概念模型：

要了解概念模型UML，首先我们需要澄清一个概念模型是什么？为什么一个概念模型是在所有需要吗？

- 概念模型可以被定义为模型，它是由概念和它们之间的关系。

- 概念模型的第一步是绘制UML图之前。它有助于了解在现实世界中的实体，以及他们如何互相交流。

UML描述的实时系统，这是非常重要的一个概念模型。 UML的概念模型可以通过学习掌握以下三大要素：

- UML构建模块
- 规则连接构建模块
- UML公共机制

面向对象的概念：

UML可以描述为面向对象的分析和设计的继任者。

一个对象包含了数据和控制数据的方法。数据表示对象的状态。类描述的对象，他们也形成层次结构模型真实世界的系统。表示为继承层次结构，也可以以不同的方式按要求相关的类。

对象是现实世界的实体存在我们周围像抽象，封装，继承，多态的基本概念，都可以使用UML表示。

因此，UML是强大到足以代表所有的概念存在于面向对象的分析和设计。 UML图是面向对象的概念的表示。因此，学习UML之前，详细了解面向对象的概念就变得非常重要。

以下是一些基本概念，面向对象的世界：

- 对象**:** 对象代表一个实体的基本构建块.
- 类**:** 类是对象的蓝图.
- 抽象化**:** 抽象代表现实世界中实体的行为.
- 封装**:** 封装是将数据绑定在一起，并隐藏他们外部世界的机制。
- 继承**:** 继承是从现有的机制作出新的类。
- 多态性**:** 定义的机制来以不同的形式存在.

面向对象的分析与设计

调查可以被定义为面向对象的分析，更具体地，它是调查对象。设计是指确定对象的协作。

所以重要的是要了解面向对象的分析和设计理念。现在，面向对象的分析的最重要的目的是要设计一个系统来识别对象。这一分析也做了为现有的系统。现在，一种有效的分析是唯一可能的，当我们能够开始思考对象可以识别的方式。确定对象后，确定它们之间的关系，并最终产生的设计。

因此，面向对象的分析与设计的目的可以描述为：

- 确定一个系统中的对象.
- 确定它们之间的关系.
- 做一个设计，使用面向对象的语言可以转换为可执行文件.

有三种基本应用面向对象的概念和实施步骤。步骤可以被定义为

```
OO Analysis --&gt; OO Design --&gt; OO implementation using OO languages
```

以上三点可以详细描述：

- 在面向对象的分析，最重要的目的是确定对象和描述他们以适当的方式。如果这些对象的有效识别，那么接下来的设计工作是很容易的。对象应确定职责。职责是对象所执行的功能。每一个对象具有某种类型的要执行的责任。当这些责任协作系统的目的达成。
- 第二阶段是面向对象的设计。在这个阶段的重点时要求及其履行情况。在这一阶段中的对象根据其预期的关联协作。协会完成设计后也完成了。
- 第三阶段是面向对象的执行。在这个阶段，设计采用面向对象语言，如Java，C++等。

UML在面向对象设计中的作用：

UML是一种建模语言，用于示范性软件和非软件系统。虽然UML用于非软件系统，重点是面向对象的软件应用建模。大多数的UML图到目前为止讨论的用于模拟静态，动态等不同的方面，如现在各方面的构件是对象。

如果我们观察到类图，对象图，协作图，交互图，将基本上基于对象的设计。

因此，面向对象的设计和UML之间的关系是非常重要的理解。根据要求，面向对象的设计转化为UML图。在详细了解UML的面向对象的概念应该学会正确。的面向对象的分析与设计完成后，下一步是很容易的。从面向对象的分析与设计的输入是输入的UML图。

UML构建模块 - UML

UML描述的实时系统，这是非常重要的一个概念模型，然后进行逐渐。 UML的概念模型可以通过学习掌握以下三大要素：

- UML构建模块
- 规则连接构建模块
- UML的公共机制

本章介绍了所有的UML构建块。 UML的构建块可以被定义为：

- 物件
- 关系
- 图

(1) 物件:

物件是最重要的UML构建块。物件可以:

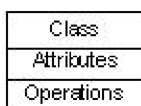
- 结构化
- 行为化
- 分组
- 注解

结构化物件:

结构性的东西定义静态模型的一部分。他们代表了物理和概念元素。以下是简要描述的结构的东西.

类：

具有类似职责的对象类表示.



接口:

接口定义了一组操作指定一个类的职责。



协作:

协作定义元素之间的相互作用。



用例:

用例代表了一组由系统的行动，为一个特定的目标。



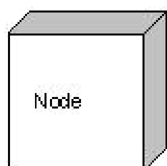
组件:

组件描述物理系统的一部分。



节点:

一个节点可以被定义为在运行时存在的物理元素。



行为物件:

行为由UML模型中的动态部分。以下是行为的东西:

交互:

交互被定义为一种行为，包括一组元素之间的消息交换来完成特定的任务。



状态机器:

状态机是有用的，当一个对象在其生命周期的状态是很重要的。它定义了一个对象的状态序列通过对事件的响应。活动负责外部因素状态变化。



组物件:

分组物件可以被定义为一种机制，一个UML模型族元素。只能有一个分组物件:

包:

封装是唯一一个组物件可收集结构和行为的东西。



注解物件:

注释物件可以被定义为一种机制来捕捉UML模型元素的言论，说明和注释。注是唯一一个注释物件可用的。

注释:

注意用于渲染意见，约束等的UML元素。



(2) 关系：

关系是另一个最重要的构建块UML。它显示元素是如何彼此相关联，此关联描述的一个应用程序的功能。

有四种可用的关系。

依赖关系:

依赖是两件事情之间的关系，其中一个元素的变化也影响到另一个。



协作:

协作基本上是一组链接UML模型元素连接。它还介绍了多少对象在这种关系中的一部分。



泛化:

泛化可以被定义为一个专门的元件连接关系与一个广义的元素。它基本上描述了在对象世界中的继承关系。



实现:

可以被定义为两个元件之间的关系，其中实现。一个元素描述了一些没有实现的责任，这和其他人实现他们。这种关系存在的情况下的接口。



(3) UML图:

UML图的整个讨论的最终输出所有要素，关系用于使一个完整的UML图，图中表示的系统。

UML图的视觉效果的整个过程中是最重要的部分。所有其他元素被用来制造一个完整的单。

UML包括以下九项图和下面的章节中描述的细节。

1. 类图
2. 对象图

3. 用例图
4. 序列图
5. 协作图
6. 活动图
7. 状态图
8. 部署关系图
9. 组件图

在本教程的后续章节中，我们将讨论所有这些图。

UML架构 - UML

任何真正的世界系统是由不同的用户使用。用户可以是开发人员，测试人员，商务人士，分析师和等等。所以在设计一个系统的体系结构是用不同的角度心态。最重要的部分是从不同的观看者的角度来看，以可视化的系统。我们更好地了解我们使系统更好。

UML定义一个系统的不同的角度起着重要的作用。这些角度是：

- 设计
- 实现
- 处理
- 部署

该中心是连接所有这四个用例视图。一个用例代表了系统的功能。因此，其他的角度连接使用的情况下。

- 系统设计包括类，接口和协作。 UML类图，对象图支持。
- 实现定义的组件组装在一起，使一个完整的物理系统。 UML组件图是用来支持实施的角度。
- 流程定义了系统的流动。因此，在设计中所用的相同的元件也可用来支持当前角度看。
- 部署代表物理节点的硬件系统构成。 UML部署图是用来支持这个角度来看。

UML建模类型 - UML

区分UML模型， UML建模用于不同类型的不同的图。有三个重要类型的UML建模：

结构建模:

系统结构建模捕捉静态功能。它们包括下列各项：

- 类图
- 对象图
- 部署图
- 包图
- 复合结构图
- 组件图

结构模型代表的系统架构，这个框架的所有其他组件存在的地方。因此，类图，组件图和部署图的部分结构建模。它们都代表的元素和机制将它们组装。

但是，从来没有的结构模型描述系统的动态行为。类图中是最广泛使用的结构图。

行为模型:

行为模型描述了在系统中的相互作用。它代表之间的交互的结构图。行为建模显示系统的动态性质。它们包括下列各项：

- 活动图
- 交互图
- 用例图

所有上述的显示在一个系统中流动的动态序列。

架构模型:

建筑模型代表了系统的总体框架。它包含了系统的结构和行为的元素。建筑模型可以被定义为整个系统的蓝图。包图是根据建筑造型。

UML基本表示法 - UML

UML是流行的图解符号。我们都知道，UML是可视化，说明，构建和记录软件和非软件系统的组成部分。这里的可视化是最重要的部分，需要被理解和记忆。

UML符号是最重要的建模元素。适当有效地使用符号是非常重要的一个完整的，有意义的模型。该模型是无用的，除非它的目的是正确描绘。

所以学习符号应该从一开始就强调。不同的符号可用于事物和关系。 UML图使用的符号物件和关系。可扩展性是另一个重要的功能，这使得UML更加强大和灵活。

本章还介绍了更详细的UML基本表示法。这仅仅是一个扩展的UML构建块段，我已经在前面的章节中讨论。

结构化物件:

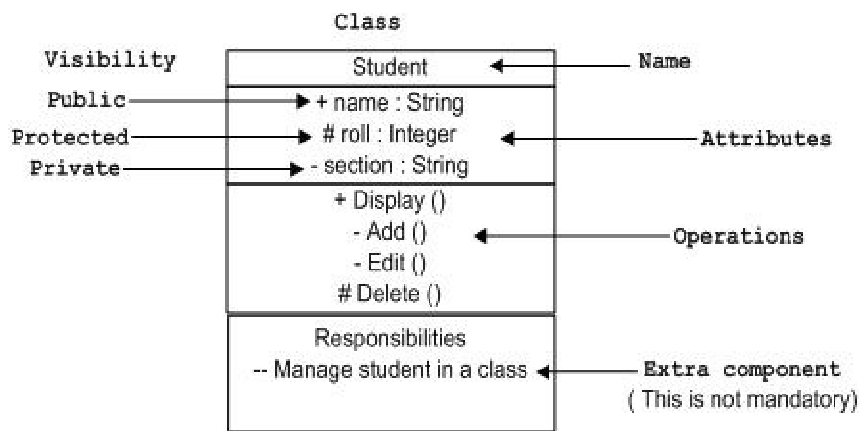
用图形表示法中使用的结构物件是UML中最广泛使用的。这些被认为是为UML模型的名词。以下是结构的东西的列表。

- 类
- 接口
- 协作
- 用例
- 活动类
- 组件
- 节点

类注释:

下面的图表示的UML类。该图被分为四个部分。

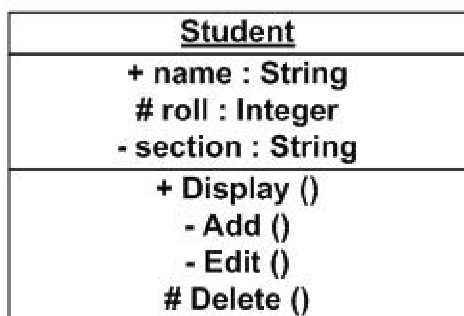
- 顶端部分被用来命名类。
- 第二个是用来显示类的属性。
- 第三部分是用来描述由类执行的操作。
- 第四部分是可选的显示附加组件。



类是用来表示对象。对象可以是任何性质和职责。

对象表示法:

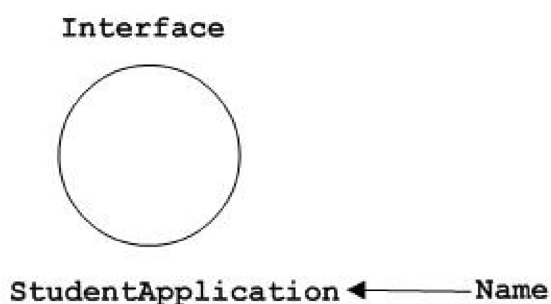
该对象表示以同样的方式作为类。唯一的区别是有下划线的名称，如下图所示。



由于对象是实际执行的一类被称为类的实例。因此，它具有相同的使用作为类。

接口表示法:

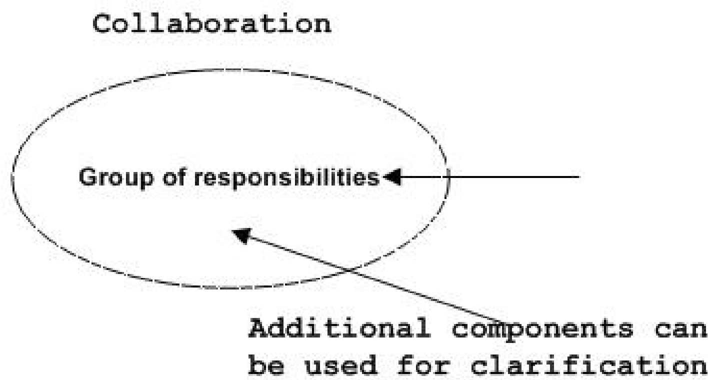
接口是用圆来表示，如下所示。它有一个名称，一般写成下面的圆圈。



接口是用来描述的功能，而不执行。界面就像一个模板，定义不同的功能不执行。当一个类实现了接口，也按要求实现的功能。

协作表示法:

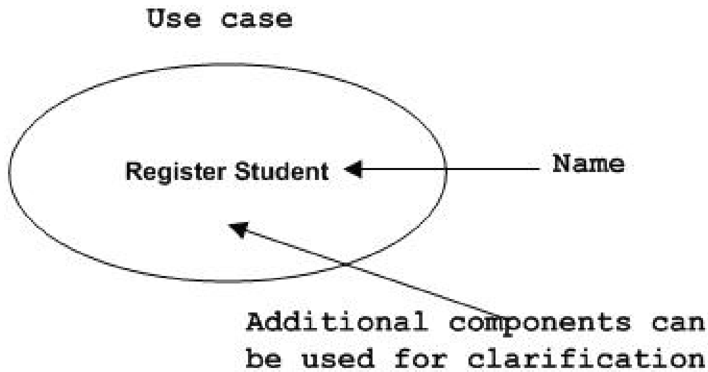
协作表示由eclipse 虚线如下所示。它有一个名字，里面写eclipse。



协作表示职责，一般职责是在一组。

用例表示法:

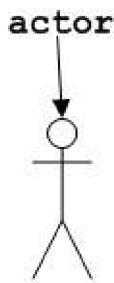
用例表示它里面的一个名字作为eclipse。它可能包含更多的责任。



用例是用来捕捉系统的高层次功能。

角色表示法:

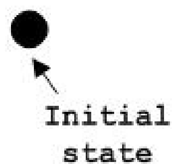
某些内部或外部的与系统进行交互的实体，可以被定义为一个角色。



角色是在用例图描述内部或外部实体。

初始状态表示法：

初始状态被定义，以显示开始的一个过程。这个符号在几乎所有的图。



初始状态的表示法的用法是显示的一个过程的起点。

最终状态表示法：

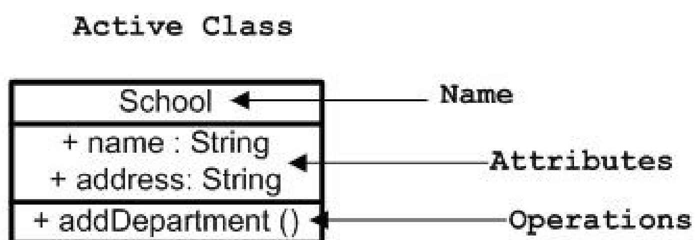
最终状态是用来显示的一个过程的结束。这种表示法也可以用来在大部分的图中描述的目的。



最终状态表示法的用法是显示一个过程的终止点。

活动类表示法：

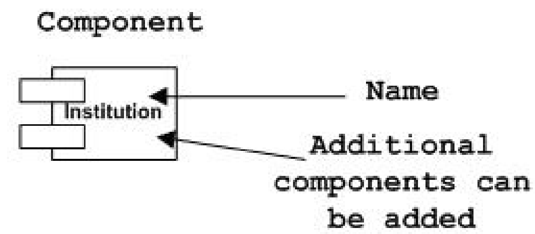
活动类类似于一类具有扎实的边界。活动类一般是用来描述一个系统的并发行为。



活动类是用来表示在一个系统的并发性。

元件表示法：

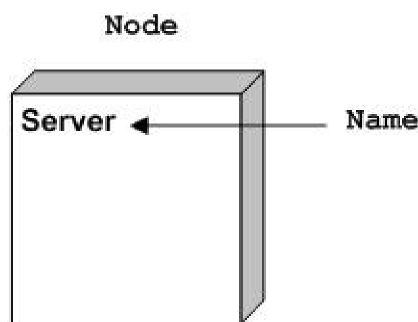
UML中的一个组件，如下图所示名称里面。在必要时，可以添加额外的元素。



元器件是用来表示系统的任何部分的UML图。

节点表示法：

UML中的一个节点表示的一个方盒子，如下图所示，同一个名字。一个节点表示一个物理的系统组件。



节点用来表示物理系统的一部分，如服务器，网络等

行为物件：

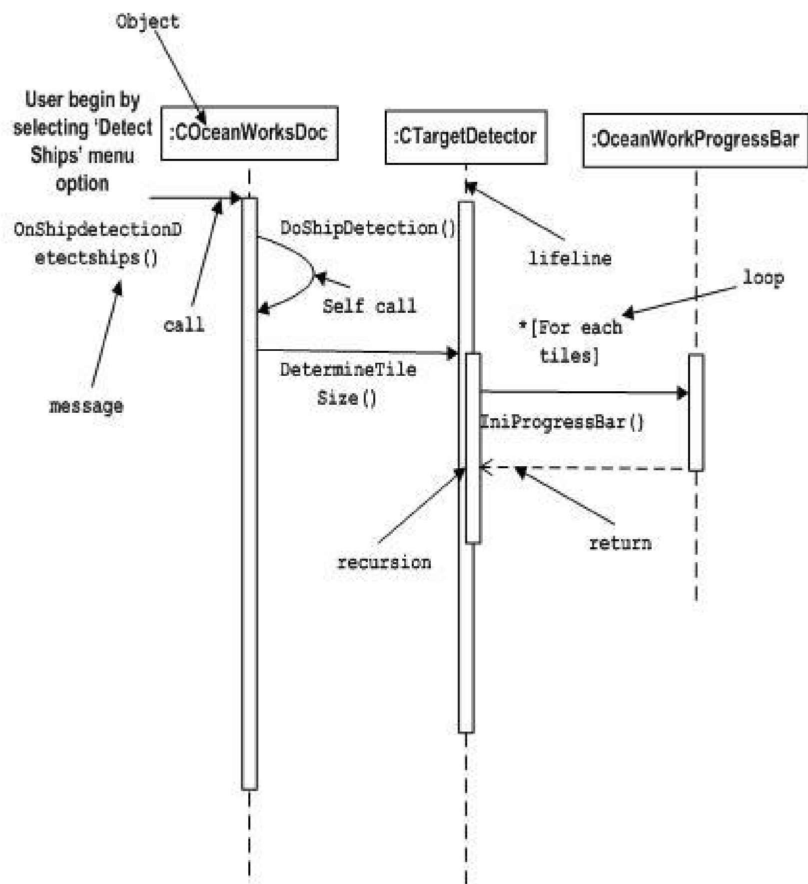
动态部分是UML中最重要的元素之一。 UML有一个强大的功能集，代表软件和非软件系统的动态部分。这些功能包括交互和状态机。

相互作用可分为两种类型：

- 顺序（序列图）
- 协作（协作图）

交互表示法：

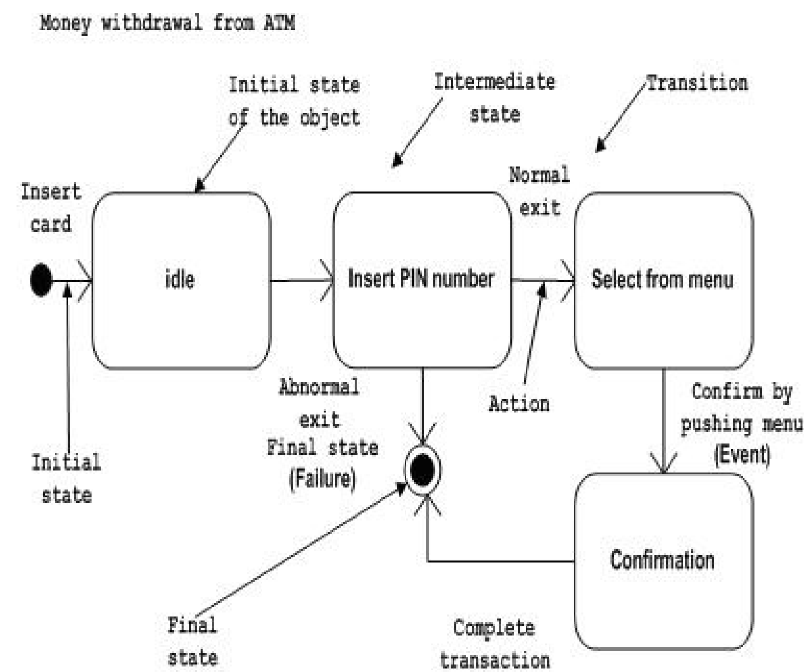
交互基本上是两个UML组件之间的信息交换。下图表示交互中使用不同的符号。



交互是用来表示一个系统的组件之间的通信。

状态机表示法：

状态机描述的组件在其生命周期的不同状态。在下面的图中描述的符号。



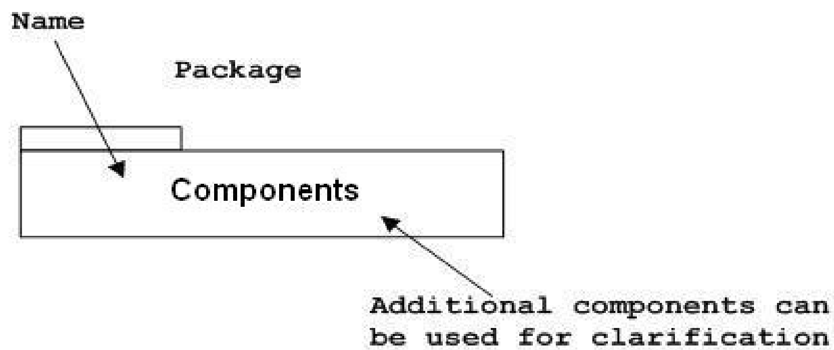
状态机是用来描述一个系统组件的不同状态。状态可以是活动，空闲或任何其他根据情况。

分组物件：

组织的UML模型设计的最重要的方面之一。 UML中只有一个元件即可用于分组，也就是包。

包表示法:

包装信息书写方式如下表所示，这是用来包装系统组成部分的。

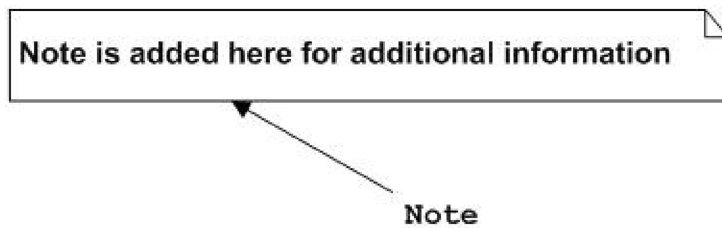


注释物件

任一图表中说明的不同的元素和它们的功能是非常重要的。因此，UML符号注释，以支持这一要求。

注释表示法:

这种表示法如下所示，它们被用来提供一个系统的必要的信息。



关系

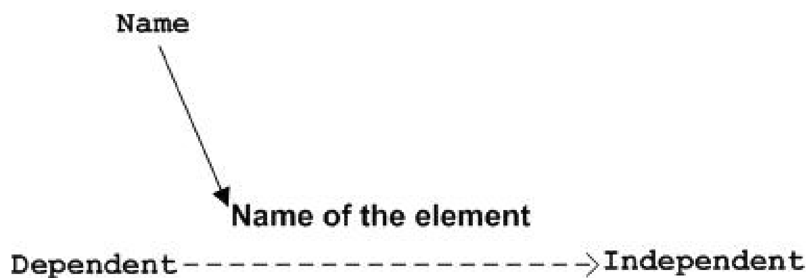
模型是不完整的，正确的描述，除非元素之间的关系。关系给出了一个UML模型的意思。以下是UML中提供了不同类型的关系。

- Dependency
- Association
- Generalization
- Extensibility

依赖表示法：

依赖是UML元素的一个重要方面。它描述了相关的元素和方向上依赖关系。

依赖关系的虚线箭头表示，如下所示。箭头代表的独立元素，另一端的依赖元素。

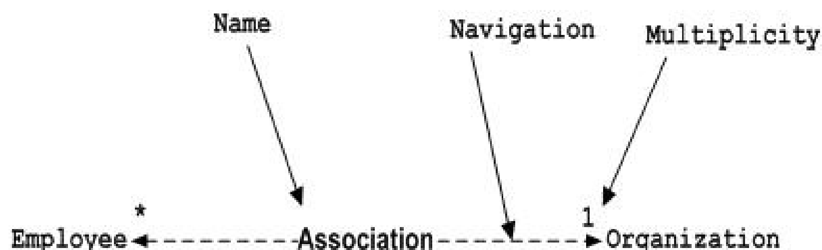


依赖关系是用来表示一个系统的两个元素之间的依赖。

协作表示法：

协作介绍UML图中的元素相关联。简单的一句话，它介绍了多少个元素参与互动。

联合会（无）两侧的箭头的虚线表示。两端代表两个相关联的元素，如下所示。在两端（1, *等）的多样性也提到多少对象相关。



协作是用来表示一个系统的两个元素之间的关系。

泛化表示法：

泛化介绍了面向对象世界的继承关系。这是父与子的关系。

泛化为代表的空心箭头，如下图所示箭头。的一端表示的父元素和子元素的另一端。

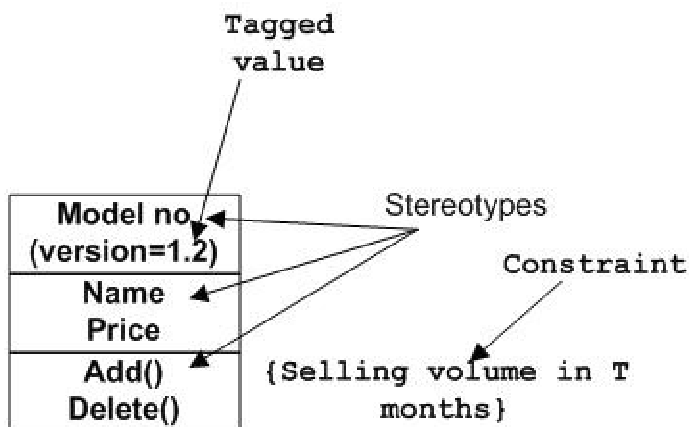


泛化是用来描述一个系统的两个元素的亲子关系。

可扩展性表示法:

所有的语言（编程或模型）有某种机制来扩展其功能类似的语法，语义等UML还具有以下机制来提供可扩展性功能。

- 定型观念(代表新元素)
- 标记值 (代表新的属性)
- 约束 (代表界限)



可扩展标记来增强语言的权力。它基本上是用来表示一些额外的系统行为的附加元素。这些额外的行为，不包括可用的标准符号。

UML标准图 - UML

在前面的章节中，我们已经讨论过的构建和其他必要的UML元素。现在，我们需要明白的地方使用这些元素。

元素都可以以不同的方式，使一个被称为图的完整的UML图片，如:组件。所以这是非常重要的，要了解不同的图表，以实现知识在现实生活中的系统。

任何复杂系统是最好的理解，通过使某种类型的图表或图片。这些图表有一个更好地影响我们的理解。所以，如果我们看看周围，那么，我们将实现图是不是一个新的概念，但它被广泛使用在不同的形式在不同的行业。

我们准备更好的和简单的方式了解一个系统的UML图。一个单一的图涵盖所有方面的制度是不够的。因此，UML定义了各种图表覆盖系统方面。

还可以创建你自己的一套图表，以满足要求。图一般都是在一个增量和迭代的方式。

有两大类的图表，分为子类：

- 结构图
- 行为图

结构图：

结构图表示的系统的静态方面。这些静态方面指示，形成的主要结构并因此稳定那些部分。

这些静态部分是表示类，接口，对象，组件和节点。四个结构图：

- 类图
- 对象图
- 组件图
- 部署图

类图：

类图是UML中使用的最常见的图。类图包括：类，接口，关联和协作。

类图，基本上代表了面向对象的视图在本质上是静态的系统。

活动类在类图来表示系统的并发性。

类图代表的面向对象的系统。因此，它一般用于开发目的。这是最广泛使用的系统架构的图。

对象图：

类图，对象图可以描述为一个实例。因此，这些图是更贴近现实生活的情况下，去实现了一个系统。

对象图是一组对象和它们之间的关系就像类图，也代表了系统的静态视图。

对象图的用法是类似的类图，但是从实际的角度来看，它们被用来建立一个系统的原型。

组件图：

组件图代表了一套组件和它们之间的关系。这些组件包括类，接口或协作。

因此，组件图表示一个系统的实现视图。

在设计阶段的软件构件（类，接口等）的系统被安排在不同的组，这取决于他们的关系。这些组被称为组件。

最后，组件图用于可视化的实现。

部署图：

部署图是一组节点和它们之间的关系。这些节点部署这些组件的物理实体。

部署图用于可视化系统的部署视图。这通常是由部署团队。

注*:* 如果上述描述和用法仔细观察，这是很清楚的，所有的图表都彼此有某种关系。组件图是依赖的类，接口等类/对象图的一部分。再次部署图是取决于使用的组件，这些组件，以使一个组件图。

行为图：

任何系统都可以有两个方面，静态和动态。因此，一个模型被认为是完成时，这两个方面都完全覆盖。

行为图基本上捕捉系统的动态方面。动态方面可以进一步改变/移动系统的一部分。

UML具有以下五种行为图：

- 用例图

- 序列图
- 协作图
- 状态图
- 活动图

用例图：

用例图是一组使用的情况下，参与者和他们的关系。他们代表了用例的系统视图。

一个用例代表一个特定的系统功能。

因此，用例图是用来描述的功能之间的关系和他们的内部/外部控制器。这些控制器是已知的参与者。

序列图：

序列图是一种交互图。从名称上很明显，图中涉及的一些序列，它是从一个对象到另一个的消息流序列。

从实施和执行的角度来看是非常重要的系统组件之间的交互。

因此，在一个系统中执行一个特定的功能的调用序列的序列图是用于可视化。

协作图：

协作图是另一种形式的交互图。它代表了一个系统的组织结构和发送/接收的消息。组织结构由对象和链接。

协作图的目的是类似的序列图。但是，协作图的具体目的是可视化的组织对象及其相互作用。

状态图：

任何实时系统预计将通过某种内部/外部事件反应。这些事件是负责对系统状态的变化。

状态图是用来表示的事件驱动的系统状态的变化。它基本上描述了类，接口状态变化等

状态图是用于可视化的反应系统内部/外部因素。

活动图：

活动图描述了在一个系统中的控制流。因此，它包括的活动和链接。流程可以是顺序，并发或分支。

活动是什么，但一个系统的功能。活动图的数字准备捕捉整个系统中的流程。

活动图用于可视化的流量控制在一个系统中。这是准备系统将如何工作，在执行时有一个想法。

注*:* 一个系统的动态性质是非常难以捕捉。因此，UML已经提供的功能，从不同的角度捕捉到的动态系统。顺序图和协作图是同构的，因此它们可以彼此转换不会丢失任何信息。这也是真实的状态图和活动图。

UML类图 - UML

概述:

类图是静态图。它代表了一个应用程序的静态视图。类图不仅用于可视化描述和记录系统的不同方面，但也为构建可执行代码的软件应用程序。

类图描述一类的属性和操作，也对系统的约束。被广泛应用于类图的建模的面向对象的系统中，因为它们是唯一的，可以直接映射到面向对象的语言的UML图。

类图显示的集合类，接口，关联，协作和约束。它也被称为作为结构图。

目的:

类图的目的是模型的一个应用程序的静态视图。类图是唯一的图可以直接映射到面向对象的语言，因此广泛应用于施工时间。

UML图，像活动图，顺序图只能给应用程序，但顺序流类图是一个有点不同。所以它是最流行的UML图编码社区。

因此，类图的目的可概括为：

- 分析和设计应用程序的静态视图。
- 描述一个系统的责任。
- 基地组件图和部署图。
- 正向和逆向工程。

如何画类图？

类图是最流行的用于建设软件应用程序的UML图。所以学习的类图的绘制过程是非常重要的。

类图有很多的属性来考虑，同时绘制，但这里的图将被视为从顶层视图。

类图基本上是一个系统的静态视图的图形表示，代表不同方面的应用。因此，集合类图表示整个系统。

画类图时，应记住以下几点：

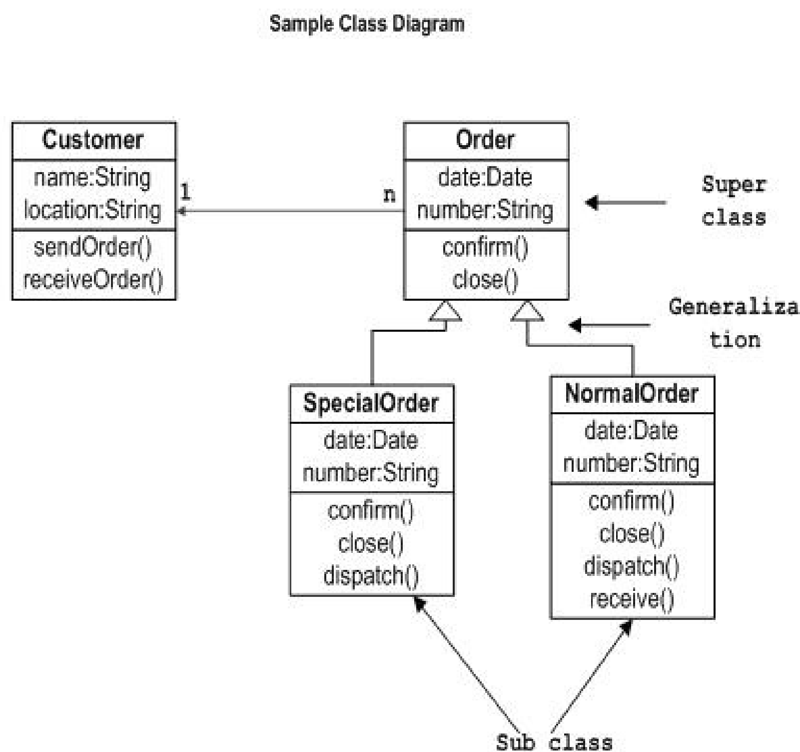
- 名称应该是有意义的类图描述面向系统。

- 应事先确定的每个元素之间的关系。
- 应清晰标明每个类职责（属性和方法）。
- 对于每个类的属性的最小数量应符合规定。因为不必要的属性将使图表复杂。
- 使用了以下注释有否要求来描述图中的某些方面。因为上面的附图，它应该是可以理解的开发者/编码器。
- 最后，在最终版本之前，该图应绘制在普通纸上尽可能多次，使其纠正和返工。

现在，下面的图是一个二阶系统的一个应用程序的一个例子。它描述了整个应用程序的一个特定方面。

- 首先所有订单及客户被认定为系统的两个要素，他们有一个一对多的关系，因为一个客户可以有多个订单。
- 我们将保持Order类是一个抽象类，它有两个具体的类（继承关系）SpecialOrder 和 NormalOrder。
- 两个继承类Order类的所有属性。此外，他们有额外的功能 *dispatch ()* 和 *receive ()*.

因此，下面的类图已经绘就考虑到所有上述提到的几点：



在哪里使用类图？

类图是一个静态图，它是用来模拟一个系统的静态视图。静态视图描述了系统的词汇。

也被认为是类图作为基础组件图和部署图。类图不仅用于可视化系统的静态视图，但它们也可用于构建可执行代码的任何系统中的前向和反向工程。

UML图一般不直接映射到任何面向对象的编程语言，但在类图是一个例外。

类图清楚地显示了映射面向对象语言，如Java，C++等，因此，从实际经验的类图通常用于构建用途。

因此，一个简短的类图用于：

- 描述系统的静态视图。
- 显示静态视图中的元素之间的协作。
- 由系统执行的功能的描述。
- 构建软件应用面向对象的语言。

UML对象图 - UML

概述:

对象图都来源于类图，依赖类图对象图。

对象图表示一个类图的一个实例。类图和对象图的基本概念是相似的。对象图也代表了一个系统的静态视图，但这种静态视图是系统在某一时刻的一个快照。

对象图是用于呈现一组对象和它们之间的关系作为一个实例。

目的:

图的目的应该清楚地理解去实现它。对象图的目的与类图类似。

不同的是，一个类图代表一个抽象的模型，包括类和它们之间的关系。但是，对象图表示在某一时刻，这在本质上是具体的实例。

这意味着对象图是更接近实际的系统行为。目的是在一个特定的时刻捕捉到静态的系统视图。

因此，对象图的目的可概括为：

- 正向和逆向工程。
- 一个系统的对象间的关系
- 一个交互的静态视图。
- 了解对象的行为和他们的关系从实用的角度来看

如何绘制对象图？

我们已经讨论过的一个对象图是类图的一个实例。它意味着一个对象图包含在类图中所用的东西的实例。

因此，这两个图均采用相同的基本元素，但在不同的形式。在类图中的元素是抽象的形式来表示蓝图，并在对象图中元素的具体形式来表示真实世界中的对象。

为了捕捉一个特定的系统，类图的数量是有限的。但是，如果我们考虑对象图，那么我们就可以有无限数量的实例在本质上是独一无二的。因此，只有这些情况下被认为是对系统的影响。

从上面的讨论，很显然，一个单一的对象图不能捕获所有必要的实例，而不能指定一个系统的所有对象。因此，解决方案是：

- 首先，分析系统，并决定哪些情况下有重要的数据和关联。
- 其次，只考虑那些实例将涵盖功能。
- 第三，做一些优化实例的数量是无限的。

绘制对象图之前，应该记住以下事情，并清楚地理解：

- 对象图是由对象。
- 对象图中的链接是用来连接对象。
- 对象和链接的两个要素，用于构造一个对象图。

在开始构建图前，现在来决定下列事项：

- 对象图应该有一个有意义的名称，以表明其目的。
- 最重要的要素是要确定。
- 对象之间的关联，应该予以明确。
- 不同元素的值需要捕获包含在对象图。
- 添加适当的注释，需要更清晰点。

下面的图是一个对象图的一个例子。它代表了订单管理系统，我们已经讨论了在类图。下图是该系统的一个实例，在一个特定的时间购买。它具有以下的对象

- 顾客
- 订单
- 特殊订单
- 一般订单

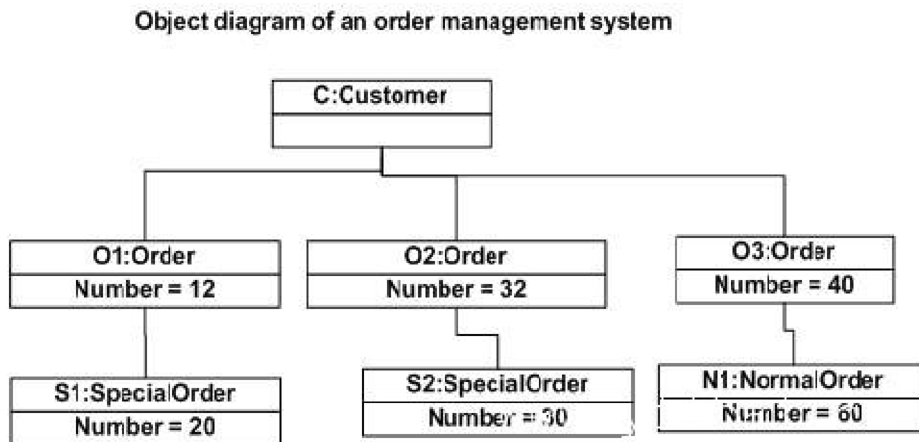
现在客户对象（C）是与三阶对象（O1，O2和O3）。这些订单对象相关联的特殊订单和一般订单对象（S1，S2和N1）。顾客具有以下三个具有不同数目的订单（12，32和40），用于所考虑的特定的时间。

现在，客户可以在将来增加的订单数量，在这种情况下对象图将反映。如果订单、特殊订单和正常秩订单对象那么观察会发现，他们有一些值。

订单的值是12，32和40，这意味着，这些对象都拥有这些实例时，捕获特定时刻的值（这里是购买时的时刻被视为特定时间）。

相同特别订订单和正常订单对象所具有的订单数分别为20，30和60。如果被认为是一个不同的时间购买，那么这些值将发生相应的变化。

因此，下面的对象图已经绘就考虑到所有上述提到的几点：



在哪里使用对象图？

对象图可以被想象成正在运行的系统在某一时刻的快照。现在加以阐明，我们可以举一个例子，一个正在运行的列车。

现在，如果把一个单元列车运行，那么会发现它具有以下静态图片：

- 这是一个特别的状态运行
- 一个特定的乘客数量。如果捕捉在不同的时间，这将在不断改变。

所以，在这里我们可以想像的列车运行的管理单元是一个对象，具有上述值。任何现实生活中的简单或复杂的系统而且的确如此。在一个简短的，对象图用于：

- 使一个系统的原型。
- 逆向工程。
- 造型复杂的数据结构。
- 从实用的角度了解系统。

UML组件图 - UML

概述:

组件图是不同的性质和行为。组件图用于模拟物理方面的系统。

现在的问题是什么，这些物理方面？物理方面的元素，如可执行文件，库，文件，证件等它位于在一个节点。

因此，组件图用于可视化的组织和系统组件之间的关系。这些图也被用来使可执行的系统。

目的:

组件图是一种特殊的UML图中。所有其他图表到目前为止讨论的目的也不同。它不描述该系统的功能，但它描述了用于使这些功能的组件。

所以从这一点来说，组件图用于可视化在一个系统中的物理组件。这些组件库，程序包，文件等。

组件图也可以被描述为一个静态的实施的系统视图。静态执行代表组织的组成部分，在一个特定的时刻。

一个单一的组件图不能代表整个系统，但图的集合可用来代表整个。

因此，组件图的目的可概括为：

- 可视化系统的组成部分。
- 构建的可执行文件，使用正向和反向工程。
- 描述的组织和组件的关系。

如何绘制组件图？

组件图是用来描述一个系统的物理构件。此神器包括文件，可执行文件，库等。

所以这张图的目的是不同的，组件图的过程中使用的应用程序的实施阶段。但它准备提前以可视化的实现细节。

最初，系统的设计使用不同的UML图，然后构件是现成的组件图是用来得到一个想法的实现。

此图是非常重要的，因为如果没有它，应用程序不能有效地实施。精心准备的组件图在其他方面也是很重要的，如应用程序的性能，维护等

所以在绘制组件图后的工件是清楚可辨：

- 在系统中使用的文件。
- 库和其他构件的申请有关。
- 构件之间的关系。

现在，确定构件需要遵循以下几点：

- 使用有意义的名称，标识组件图要绘制。
- 作好心理准备之前的布局使用的工具。
- 使用说明明确的要点。

下面是一个订单管理系统的组件图。这里的构件是文件。所以，该图显示了在应用程序的文件和它们之间的关系。在实际组件图还包含dll文件，库，文件夹等。

在下面的图中，四个文件识别，并产生了它们之间的关系。到目前为止讨论与其他UML图，组件图不能直接匹配。因为它是得出完全不同的目的。

所以下面的组件图已经绘就考虑到所有上述提到的几点：



在哪里使用组件图？

我们已经描述组件图用于可视化系统的静态实现视图。组件图是特殊类型的UML图，但用于不同的目的。

这些图显示系统的物理组件。要澄清，我们可以说，组件图描述了在一个系统中的组件组织。

组织机构可以进一步描述为在一个系统中的组件的位置。这些组件是在一个特殊的组织方式，以满足系统要求。

正如我们已经讨论过这些组件库，文件，可执行文件等，现在组织实施前的应用程序，这些组件。此组件组织还单独设计作为项目执行的一部分。

从执行的角度来看，是非常重要的组件图。因此，应用程序的执行团队应该有一个正确的认识组件的详细信息。

载入组件图的使用可以被描述为：

- 组件建模的一个系统。

- 模型的数据库架构。
- 模型的应用程序的可执行文件。
- 模型系统的源代码。

UML部署图 - UML

概述:

部署图用于可视化的软件组件部署的系统中的物理组件的拓扑结构。

因此，部署图是用来描述一个系统的静态部署视图。部署图由节点和它们之间的关系。

目的:

部署名称本身描述的原理图的目的。部署图用于描述软件组件部署的硬件组件。组件图和部署图是密切相关的。

组件图是用来描述的组件和部署图显示了它们是如何在硬件中部署。

UML的设计主要是把重点放在系统的软件构件。但是，这两个图是使用特殊图表专注于软件组件和硬件组件。

所以大多数的UML图是用来处理逻辑组件，但把重点放在系统的硬件拓扑部署图。部署图用于由系统工程师。

部署图的目的，可以描述如下：

- 可视化系统的硬件拓扑。
- 描述用于部署软件组件的硬件组件。
- 描述运行时处理节点。

如何绘制部署图？

部署图部署的系统视图。据相关的组件图。由于组件的部署使用的部署图。部署图由节点。节点是什么，但用于将应用程序部署的物理硬件。

部署图对系统工程师是非常有用。一个高效的部署图是非常重要的，因为它控制以下参数

- 性能
- 可扩展性
- 可维护性
- 可移植性

因此，绘制部署图前应确定以下构件：

- 节点
- 节点之间的关系

下列部署图是一个样品给订单管理系统的部署视图的想法。在这里，我们已经表明节点：

- 监控
- 调制解调器
- 缓存服务器
- 服务器

假定应用程序是一个基于Web的应用程序部署在集群环境中使用服务器1，服务器2和服务器3。用户连接到使用互联网的应用程序。控制流从缓存服务器的集群环境中。

所以下面的部署图已经制定考虑到所有上述提到的几点：



在哪里使用部署图？

部署图主要用于系统工程师。这些图用来描述的物理组件（硬件），它们的分布和关联。

为了阐述清楚细节，我们可以想像的硬件组件/节点上的软件组件位于部署图。

软件应用程序的开发复杂的业务流程模型。只有高效的软件应用是不够的，以满足业务需求。业务需求可以被描述为支持不断增长的用户数，响应时间快等

为了满足这些要求的硬件组件的类型应该被设计效率和以具有成本效益的方式。

当前软件应用程序在本质上是非常复杂的。软件应用程序可以是独立的，基于Web，分布式，基于大型机和许多更多。所以这是非常重要的，以有效地设计的硬件组件。

因此，使用部署图可以描述如下：

- 为了模拟一个系统的硬件拓扑。
- 嵌入式系统建模。
- 为了模拟一个客户机/服务器系统的硬件的详细信息。
- 为了模拟硬件的分布式应用程序的细节。
- 正向和逆向工程。

UML用例图 - UML

概述:

为了模拟系统最重要的方面是捕捉到的动态行为。为了阐明位详细信息，动态的行为意味着它运行时/操作系统的行为。

因此，只有静态的行为是不够的模拟系统，而动态的行为，更重要的是比静态行为。在UML模型的动态性质和使用情况图5图就是其中之一。现在我们要讨论的，本质上是动态的用例图，应该有一些内在或外在因素互动。

这些内部和外部代理是已知的行为体。因此，用例图由主角，用例和它们之间的关系组成。该图是用来模型的一个应用程序的系统/子系统。一个单一的用例图捕获系统的特定功能。

因此，来模拟整个系统的用例图。

目的:

用例图的目的是捕捉到一个系统的动态方面。但这一定义过于笼统描述其目的。

因为其他的四个图解的图（活动，序列，协作和状态图）也具有同样的目的。因此，我们将寻找到一些特定的目的，这将区别于其他四幅图。

用例图是用来收集系统的要求，包括内部和外部的影响。这些要求大多是设计要求。所以，当一个系统的分析，以收集其功能用例和参与者确定。

现在，当最初的任务是完整的用例图是仿照目前外界的视图。

因此，简言之，用例图的目的如下：

- 用来收集系统的要求。
- 用于获取系统的外观图。
- 识别外部和内部因素影响系统.
- 显示要求之间的相互作用是参与者.

如何画用例图？

用例图被认为是高层次的需求分析系统。因此，当系统的要求，分析被捕获在用例的功能。

因此，我们可以说，使用情况是什么，但在一个有组织的方式编写的系统功能。现在到用例相关的第二件事情是参与者。行为者可以被定义为与系统进行交互的东西。

参与者可以是人的用户，一些内部的应用程序，或可能会有一些外部应用程序。因此，在一个简短的，当我们正计划绘制一个用例图中应该有以下项目:

- 功能被表示为一个用例
- 参与者
- 用例和参与者之间的关系。

绘制到用例图捕获系统的功能要求。因此，确定上述项目后，我们必须遵循以下指导原则，绘制一个有效的用例图。

- 一个用例的名称是非常重要的。所以名的选择应以这样的方式，以便它可以识别执行的功能。
- 给出一个合适的名参与者。
- 图中清楚地显示关系和依赖性。
- 不要试图包括所有类型的关系。由于该图的主要目的是确定要求。
- 使用注意以往任何时候都需要阐明一些重要的点。

下面是一个示例用例图，代表订单管理系统。因此，如果我们看看图，那么我们会发现三个用例（订单，特殊订单和正常订单）和一个参与者：顾客。

SpecialOrder 和*NormalOrder* 从订单使用情况扩展。因此，他们扩展了关系。另外很重要的一点是确定系统边界，这是图中所示。参与者是客户以外的系统，因为它是系统的外部用户。



在哪里使用用例图？

正如我们已经讨论过，有五个在UML图建模系统的动态视图。现在，每个模型有一些特定目的使用。其实这些特定的目的，是正在运行的系统中的不同角度。

所以要了解一个系统的动态，我们需要使用不同类型的图表。用例图就是其中之一，其具体目的是收集系统的的需求和参与者。

用例图指定系统的事件和他们的流向。但从未用例图描述了他们是如何实现的。可以被想象成一个黑盒子，只有输入，输出和黑盒子的功能被称为用例图。

在这些图中使用的设计在一个非常高的水平。那么这种高层次的设计高雅，一遍又一遍完善使系统得到一个完整实用的图片。一个结构良好的用例，还介绍了前置条件，后置条件和例外。而这些多余的元素在执行测试时被用来制造测试的情况下。

用例都不是正向和反向工程，但他们仍然使用略有不同的方式。同样是真实的逆向工程。仍用例图的使用方式不同，使其逆向工程的一个候选。

在正向工程用例图是用来做测试案例和逆向工程中的使用情况下是用来准备从现有的应用程序的需求细节。

所以下面的地方使用用例图：

- 需求分析和高水平的设计。
- 模拟系统的上下文。
- 逆向工程。
- Forward engineering.

UML交互图 - UML

概述:

从名字交互作用很明显，图中是用来描述一些不同的模型中的不同元素之间的相互作用。所以，这种相互作用是动态行为的系统的一部分。

这种互动行为表示UML中的两个图，被称为序列图和协作图。这两个图的基本宗旨是相似的。

序列图强调时间顺序的消息和协作图注重发送和接收消息的对象的组织结构。

目的:

交互图的目的是可视化系统的交互行为。载入可视化的交互是一个困难的任务。因此，解决方案是使用不同类型的模型来捕获不同方面的相互作用。

这就是为什么序列和协作图是用来捕获动态性质，但是从不同的角度。

因此，交互图而言，可以描述为：

- 捕捉一个系统的动态行为。
- 来描述该系统中的消息流。
- 来描述对象的结构组织。
- 为了描述对象之间的互动。

如何绘制交互图？

正如我们已经讨论交互图的目的是捕捉系统的动态环节。因此，动态捕捉方面，我们需要了解一个动态的环节是，它是如何可视化。动态方面可以定义为在一个特定的时刻运行的系统快照。

我们有两种类型UML交互图。一个是序列图，另一种是在协作图。序列图捕获从一个对象到另一个的时间顺序的消息流和协作图描述系统中对象的组织参加在消息流中。

因此，下面是确定之前绘制交互图：

- 参与互动的对象。
- 对象之间的消息流。

- 消息的顺序流程。
- 对象的组织。

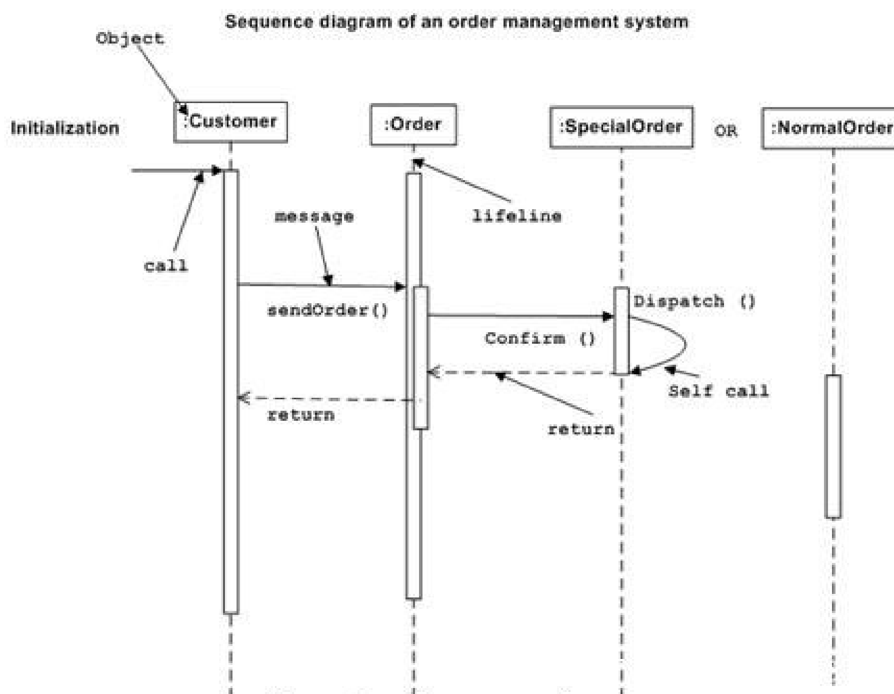
以下是两个交互图建模的订单管理系统。第一图是一个顺序图，第二个是在协作图。

序列图：

序列图有四个对象（客户，订单，特殊订单和正常订单）。

下面的关系图所示的消息序列为SpecialOrder对象和NormalOrder对象在相同的情况下使用。现在重要的是要了解的时间顺序的消息流。消息流无关，但一个对象的方法调用。

首先调用的是sendOrder（），这是一个订单对象的方法。在下一次调用*confirm ()*，这是一个方法SpecialOrder对象的最后调用*Dispatch ()*，它是一种方法的SpecialOrder对象。所以这里的图主要描述的方法调用从一个对象到另一个，在系统运行时这也是实际情况。

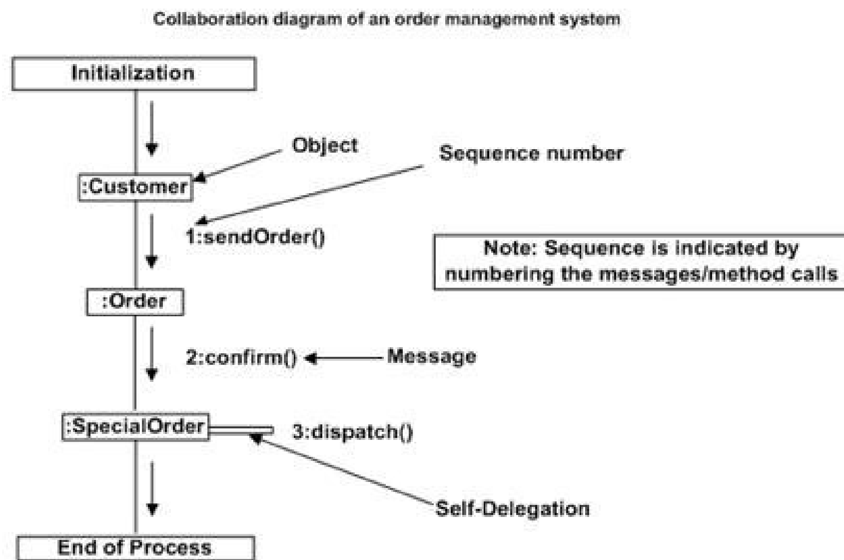


协作图：

第二交互图，协作图。它显示的对象组织，如下所示。在这里，在协作图的方法调用序列是表示，由一些数字技术，如下所示。该数字表示方法如何被称为此起彼伏。我们已经采取了相同的订单管理系统，协作图来描述。

这些调用方法类似的序列图。但不同的是，序列图中未介绍的对象组织，而协作图中示出的对象的组织。

现在选择这两个图表之间主要强调的是需求类型。如果时间序列是很重要的，那么序列图中被使用，并且，如果需要的组织，那么使用协作图。



在哪里使用交互图？

我们已经讨论了交互图是用来描述一个系统的动态本质。现在，我们将进入实用化的情况下，使用这些图。要了解实际应用中，我们需要了解的基本性质顺序图和协作图。

这两个图的主要目的，是相似的，因为它们是用来捕捉系统的动态行为。但具体的目的，更重要的是阐明和理解。

序列图是用来捕获从一个对象到另一个消息流的顺序。和协作图用来描述参与相互作用中的对象的结构组织。一个单一的图是不是足以说明整个系统的动态环节，这样的一套图是用来捕获的是作为一个整体。

使用交互图，当我们想要了解的消息流和组织结构。消息流装置控制流从一个对象到另一个序列和结构组织的装置，在一个系统中的元素的视觉组织。

在一份简短的以下交互图的用法：

- 按时间顺序的控制流建模。
- 为了模拟流结构组织控制。
- 对于正向工程。
- 逆向工程。

UML状态图 - UML

概述:

图表本身的名称，阐明该图的目的和其他细节。它描述了在一个系统中的一个组成部分不同的状态。状态是特定的一个系统的组件/对象。

状态图描述了一个状态机。我们阐明的状态机可以被定义为一台机器，它定义了一个对象，这些状态控制的外部或内部事件的不同状态。

在下一章节解释的活动图，状态图是一种特殊的。作为状态图定义了状态，它被使用的对象的生存期模型。

目的:

状态图是一个用于模拟系统的动态性质的五个的UML图。他们定义一个对象在其生命周期的不同状态。这些状态改变的事件。因此，状态图是有用的模型反应系统。反应式系统可以被定义为一个系统，响应外部或内部事件。

状态图描述从一个状态到另一个状态的控制流。国被定义为一个条件在其中一个对象存在，它改变一些事件被触发时。所以最重要的目的是状态图模型对象从创建到终止的生命周期。

状态图也可用于一个系统的前向和反向工程。但主要目的是为了模拟响应系统。

以下是使用状态图的主要目的：

- 为了模拟系统的动态环节。
- 反应系统模型生命周期。
- 一个对象来描述不同的状态，在其生命周期的时间。
- 定义一个状态机模型状态的对象。

如何绘制状态图？

状态图是用来描述不同的对象在其生命周期的状态。因此，强调的是一些内部或外部事件的状态发生变化时。这些对象的状态是重要的分析和准确的贯彻落实。

状态图描述的状态是非常重要的。对象的状况，当发生特定事件时，可以被确定为状态。

绘制状态图之前，我们必须明确以下几点：

- 识别对象，以进行分析。
- 识别状态。
- 识别的事件。

下面是一个例子，一个订单对象的状态的状态图分析。

第一个状态是空闲状态的过程从哪里开始。接下来的状态到达的事件，如发送请求，确认请求，并调度顺序。这些事件负责订单对象的状态变化。

在对象的生命周期（这里为了对象）通过以下状态，并有可能也存在一些不正常的。这种不正常的退出，可能会出现由于系统中的一些问题。整个生命周期完成时，它被视为完整的交易下文所述。

一个对象的初始状态和最终状态也如下所示。



在哪里使用状态图？

从上面的讨论中，我们可以定义一个状态图的实际应用。状态图是用来模拟动力系统环节，像其他在本教程中废弃不用四幅图。但它也有一些显着特征建模动态特性。

状态图定义了一个组件的状态，这些状态的变化在本质上是动态的。因此，其具体目的是定义由事件触发的状态变化。事件是系统的内部或外部的影响因素。

使用状态图模型状态和系统上运行的事件。当实现一个系统，这是非常重要的，以阐明在其生命周期的时间和状态图是用于此目的的一个对象的不同状态。当这些状态和事件识别它们被用来建模和制度的实施过程中使用这些模型。

如果我们看一下然后进入实际执行状态图，它主要是用来分析受事件影响的对象状态。这种分析是有帮助的，在其执行过程中了解系统行为。

因此，主要的用法可以被描述为：

- 为了模拟一个系统的对象的状态。
- 为了模拟响应系统。反应体系由反应物。
- 为了找出事件负责任的状态变化。
- 正向和逆向工程。

UML活动图 - UML

概述:

活动图是另一个重要的UML图来描述系统的动态方面。

活动图基本上是代表流程形成一个活动到另一个活动的流程图。活动可以被描述为一个系统的操作。

因此，绘制控制流从一个操作到另一个。此流可以是连续的，支链的或同时的。活动图处理所有类型的流程控制，通过使用不同的元素，如交叉、加入等

目的:

活动图的基本用途是其他四个图类似。它能够捕捉到该系统的动态行为。其他四幅图是用来显示从一个对象到另一个消息流，但用来显示消息流从一个活动到另一个活动图。

活动是一个特别的系统的操作。活动图不仅用于可视化系统的动态性质，但它们也可用于通过使用正向和逆向工程技术来构建可执行的系统。唯一缺少的东西在活动图的消息部分。

它并不显示任何消息流程从一个活动到另一个。活动图是一段时间视为流程图。虽然图中看起来像一个流程图，但事实并非如此。它显示不同的流程，如并行，分支，并发单。

因此，目的可以被描述为：

- 绘制活动流程系统。
- 描述的顺序从一个活动到另一个。
- 描述系统并行，分支，并发流。

如何绘制活动图？

活动图主要用于为流程图包括由系统执行的活动。但活动图是不完全的，因为他们有一些额外的功能流程图。这些额外的功能，包括分支，平行流，泳道等

之前绘制活动图，活动图中使用的元素，我们必须有一个清醒的认识。活动图的主要元素是活动本身。一个活动是由系统执行的功能。确定活动后，我们需要了解他们是如何相关的约束和条件。

所以在绘制活动图，我们应该确定以下要素：

- 活动
- 交互
- 条件
- 约束

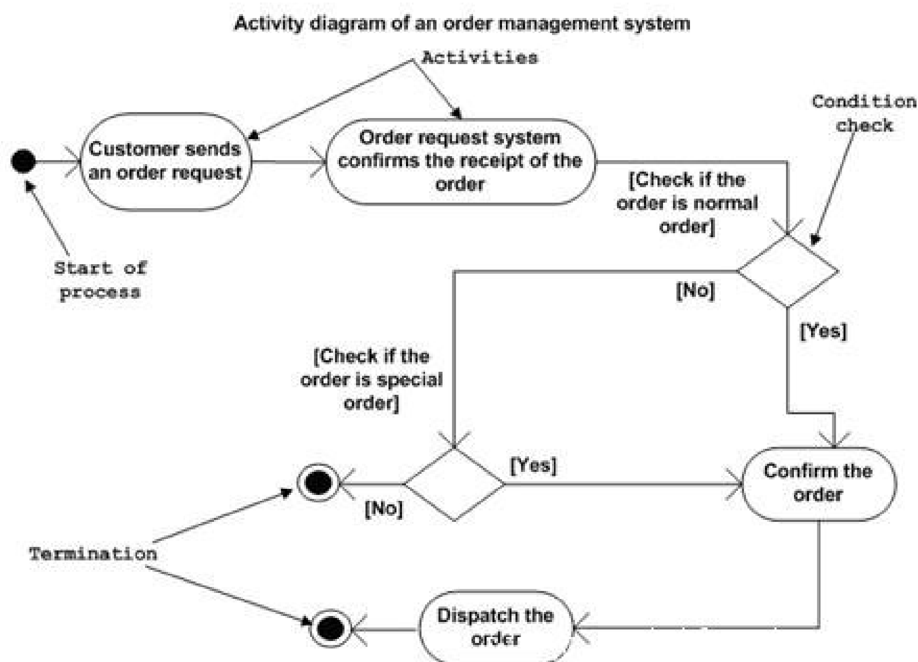
上述参数确定后，我们需要做一个心理布局整个流程。这种心理的布局转化成一个活动图。

下面是一个例子，订单管理系统的活动图。在图中确定了四个活动都与条件。其中重要的一点应该清楚地了解活动图不能完全匹配的代码。活动图了解活动流程，主要用于企业用户。

下图绘制的四个主要活动：

- 由客户发送订单
- 收到订单
- 确认订单
- 分发订单

收到订单后请求状态进行检查，以检查它是否是正常的或特殊的顺序。不同的顺序确定之后，执行调度活动，并标记为终止进程。



在哪里使用活动图？

活动图的基本用法是类似于其他四个UML图。的具体用法是模拟控制流从一个活动到另一个。该控制流程不包括消息。

活动图是适用于该系统的活动流程建模。应用程序可以有多个系统。活动图也抓住了这些系统，并介绍了流程从一个系统到另一个。在其他图中，这个特定的用法，不提供。这些系统可以是数据库，外部队列或任何其他系统。

现在，我们将看看活动图到实际应用。从上面的讨论，很显然，活动图是来自一个非常高的级别。因此，它给出了一个系统的高级视图。这种高层次的观点主要是针对企业用户或任何其他人而不是一个技术人员。

此图是用来模拟活动却都业务需求。因此，图有业务的理解，而实施细节上更具冲击力。

以下是活动图的主要用途：

- 使用业务建模工作流程。
- 建模的业务需求。
- 高层次的理解系统的功能。
- 调查在后一阶段的业务需求。

UML快速指南（摘要） - UML

UML 概述:

UML是一个通用的建模语言。它最初开始捕捉到复杂的软件和非软件系统的行为，现在它已经成为一个OMG标准。

UML提供元素和组件的复杂系统支持的要求。 UML遵循面向对象的概念和方法。因此，面向对象的系统通常使用的图案语言建模。

UML图绘制等从不同的角度设计，实现，部署等

上面的结论UML可以被定义为一种建模语言，捕捉到一个系统的体系结构，行为和结构层面。

对象是这个面向对象世界的关键。面向对象的分析和设计的基本要求，是有效地识别对象。责任分配给对象。一旦这个任务完成了设计使用输入分析。

UML具有重要的作用，在该面向对象的分析与设计，用于模拟设计的UML图。因此，UML有一个发挥重要作用。

UML 注释:

UML符号是最重要的建模元素。适当有效地使用符号是非常重要的一个完整的，有意义的模型。该模型是无用的，除非它的目的是正确描绘。

所以学习表示法应该从一开始就强调。不同的符号可用于物件和关系。 UML图使用的表示法事物和关系。可扩展性是另一个重要的特点，这使得UML更加强大和灵活。

UML图:

图表UML的核心。这些图是大致归类为结构和行为图。

- 结构图是由静态图，如类图，对象图等
- 行为图是由像序列图，协作图等动态图

一个系统的静态和动态特性是通过使用这些图的可视化。

类图:

类图是使用面向对象的社会最流行的UML图。它描述了在一个系统中的对象和他们的关系。类图包含的属性和功能。

一个单独的类图描述系统的一个具体方面，收集类图表示整个系统。基本上，类图表示系统的静态视图。

类图是唯一可以直接映射到面向对象的语言UML图。因此，它被广泛应用于开发者社区。

对象图：

对象图是类图的一个实例。因此，一类图的基本要素是类似的。对象图是由对象和链接。在一个特定的时刻，它捕获该系统的实例。

对象图用于原型设计，逆向工程和实际场景建模。

组件图：

组件图是一种特殊的UML图来描述系统的静态实现视图。组件图包括物理组件，如库，档案，文件夹等。

此图是用来从实施的角度。使用一个以上的元件图来表示整个系统。正向和逆向工程技术的使用，使可执行文件组件图。

部署图：

组件图是用来描述一个系统的静态部署视图。这些图主要用于系统工程师。

部署图是由节点和它们之间的关系。一个高效的部署图是应用软件开发的一个组成部分。

用例图:

用例图是用来捕捉系统的动态性质。它由使用的情况下，参与者及其相互关系。一个高层次的设计用例图是用来捕捉系统的要求。

因此它代表系统的功能和流向。虽然用例图的正向和反向工程是不是一个很好的选择，但他们仍然在一个稍微不同的方法来模拟它。

交互图：

交互图，用于捕获系统的动态性质。顺序图和协作图，交互图用于此目的。

序列图是用来捕获时间顺序的消息流和协作图是用来了解系统的组织结构。一般一组序列和协作图用于模拟整个系统。

状态图：

状态图是一个用于模拟系统的动态性质的五个图。这些图用来模拟一个对象的整个生命周期。活动图是一种特殊的状态图。

一个对象的状态被定义为对象所在的条件下，特定的时间和对象移动对其他状态，在某些事件发生时。状态图还用于正向和反向工程。

活动图：

活动图是另一个重要的动态行为图来描述。活动图由活动环节，关系等模型所有类型流，如平行的，单一的，并发等

活动图描述了流程控制，从一个活动到另一个无需任何消息。使用这些图的业务需求建模的高级视图。

UML 2.0 - UML

概述:

在世界上统一建模语言UML2.0是完全不同的维度。它在本质上更加复杂和广泛。

与UML1.5版本相比，文件的程度也增加了。 UML2.0中还增加了新的功能，所以它的使用可以更广泛。

UML2.0将正式和完全定义语义的定义。这种新的可能性可以用于模型的开发，并从这些模型可以产生相应的系统。但要利用这个新的层面，必须作出相当大的努力，获得知识。

UML2.0的新的层面：

UML的结构和文档UML2.0的最新版本进行了全面修订。现在有两个文件，描述UML：

- UML2.0架构的定义是基于UML语言的基本结构。本节是UML的用户并不直接相关。这是指向对建模工具的开发。所以，这方面不是在本教程的范围。
- UML2.0上盖定义UML2.0的用户结构。这意味着这些用户将立即使用的UML元素。因此，这是UML的用户群体的主要焦点。

这个版本的UML创建完成一个目标，调整和完善UML，以便简化可用性，实施和适应。

使用UML基础设施：

- 提供了一个可重用的元语言的核心。这是用来定义UML本身。
- 提供机制调整的语言。

使用UML上层建筑：

- 基于组件的发展提供更好的支持。
- 提高架构规范构造。
- 提供更好的选择行为建模。

所以很重要的一点要注意的是上述的主要分部。这些区划是用来增加UML的可用性和定义清楚地了解它的用法。

另外一个方面，已经提出了这个新版本。它是一个完全新的对象约束语言（OCL）和图交汇处的建议。这些功能都一起形成完整的UML2.0包。

UML2.0建模图：

建模的相互作用：

UML2.0中描述的交互图是比旧版有所不同。但基本概念是一样的早期版本。主要的区别是增强和附加功能添加到UML2.0图。

UML2.0模型对象在以下四个不同的方式互动。

- 序列图中的对象之间的交互来完成，系统的行为目标是一个随时间变化的图。时间序列是类似于早期版本的序列图。在系统内的设计上的交互，可以在任何级别的抽象设计，从子系统交互的实例级。
- 通信图是UML2.0中添加一个新的名字。通信图是对象之间的消息传递，协作图UML1.4和更早的版本概念的结构图。这可以定义为协作图的修改版本。
- 此外，在UML2.0也是一个新的互动概述图。一组组合成一个逻辑顺序的相互作用，包括流量控制逻辑之间的互动导航的互动概述图描述了一个高层次的。
- 时序图中还增加了UML2.0。这是一个可选的设计的一个交互的过程中发送和接收的消息中指定的时间限制的图。

因此，从上面的描述中，重要的是要注意，所有的图的目的是发送/接收消息。载入这些消息的装卸内部的对象。所以对象也有接收和发送邮件的选项，这里谈到的另一个重要方面称为接口。现在，这些接口是负责接受和发送消息到另一个。

因此，从上面的讨论可以得出结论，UML2.0中相互作用以不同的方式描述的，这就是为什么进入图片所遇到的新的图名。但是，如果我们分析了新的图，那么很显然，根据在早期版本中所描述的交互图创建的所有图。唯一的区别是UML2.0添加附加功能。使图更高效和目的导向。

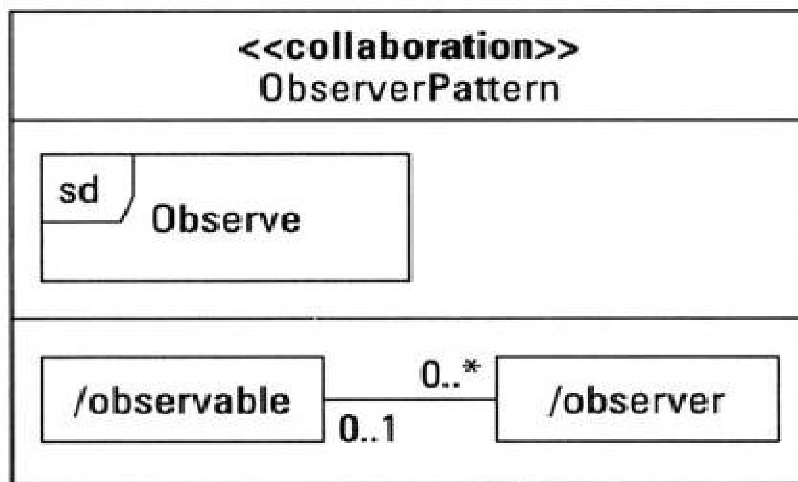
建模协作：

正如我们已经讨论过的，协作是用来模拟常见的物体之间的相互作用。要阐明的话，我们可以说，协作是互动对象由一组消息预先定义的角色。

最重要的一点要注意的是协作图的早期版本，并在UML2.0版本之间的差异。因此，区分协作图名称已更改于UML2.0。它被命名为UML2.0通信图。

因此，协作被定义为一类的属性（属性）和行为（操作）。的协作类上的隔间可以用户定义的也可用于相互作用（时序图）的构成要素（组合结构图）。

下图模型的观察者设计模式之间的协作对象观察到的项目中的作用，以及任何数量的观察员的对象。



建模通信：

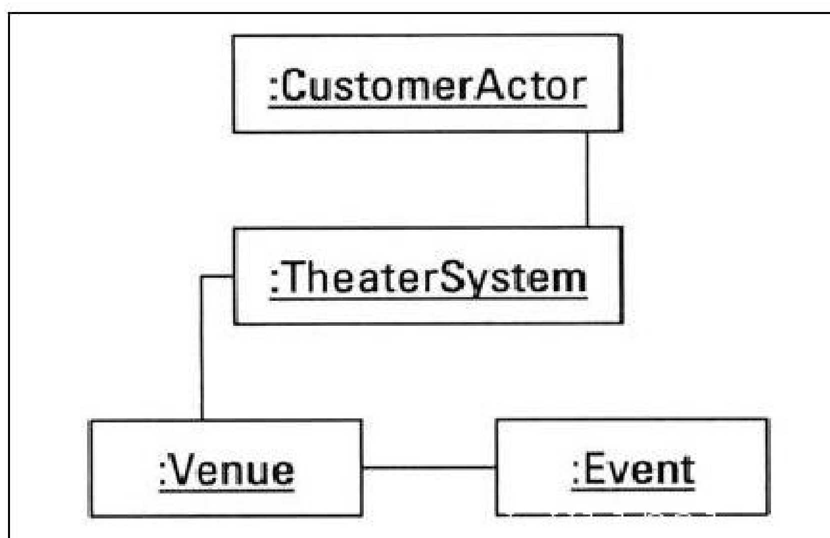
通信图协作图的早期版本略有不同。我们可以说，它是一个缩减版的早期版本的UML。通信图的区别因素是在对象之间的链接。

这是一个可视化的链接，它缺少的序列图。在序列图只显示对象之间传递的消息，即使有它们之间没有联系。

通信图是建模人员是用来防止这样的错误，通过使用一个对象图的格式作为消息传递的基础。通信图上每个对象被称为对象生命线。

通信图的消息类型是相同的序列图。通信图可以模拟同步，异步，返回，丢失，发现，和对象的创建消息。

下图显示了三个对象的对象图和两个环节，形成了基础通信图是。通信图是上每个对象被称为对象生命线。



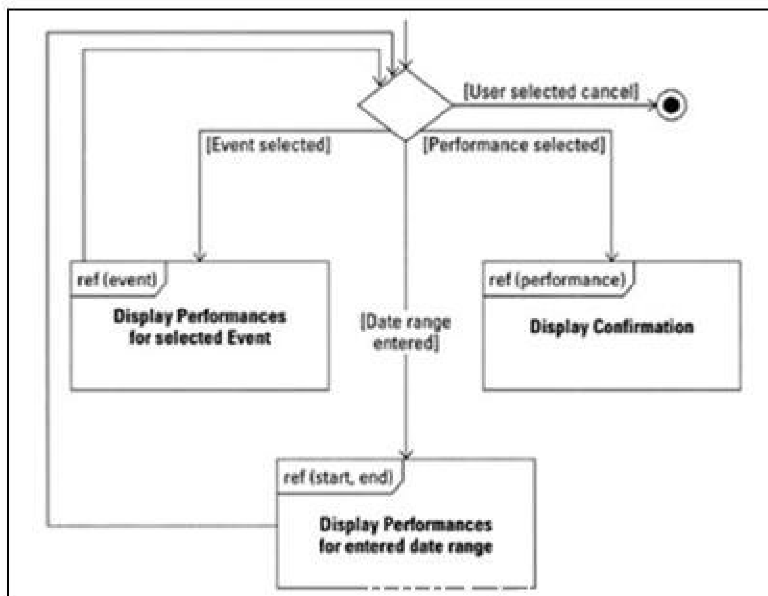
建模互动概述：

在实际使用中，一个单一的场景的序列图是用来模型。所以使用序列图来完成整个应用程序。当一个单一的场景建模，它有可能忘记的全过程并且这可能带来误差。

因此，要解决这个问题，新的互动概述结合的控制流图，活动图，序列图和消息规范。

活动图使用活动对象流来形容一个过程。互动概述图使用相互作用和交互出现。序列图中的生命线和消息只出现内相互作用或相互作用的发生。然而，参与的互动概述图的生命线（对象）可能被列为图名。

下图显示了一个决定帧和终止点的交互概览图



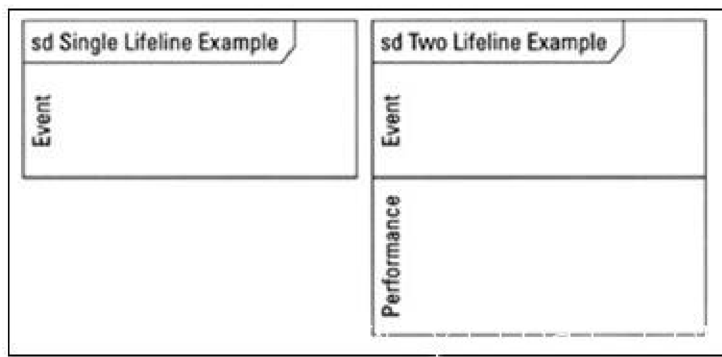
建模时序图：

此图中本身的名称，描述图中的目的。它基本上是涉及在其整个生命周期中的事件的时间。

因此，可以被定义为一个时序图，把重点放在其使用寿命中的一个对象的事件的特殊目的的交互图。它基本上是一个混合的状态机和交互图。时序图使用下面的时间线：

- 状态的时间线
- 一般值的时间线

在时序图中的生命线一帧的内容区域内形成一个长方形的空间。它通常是水平对齐读取由左到右。在同一帧内，也可以层叠多个生命线，它们之间的相互作用模型。



总结：

UML2.0是一个增强版本的新功能被添加到使它更可用，高效。在UML2.0的主要有两大类，一个是UML超级结构和另一个是UML基础设施。虽然新的图表是基于旧的观念，但他们仍然有额外的功能。

UML2.0提供了四个交互图，序列图，通信图，交互概览图，和一个可选的时序图。所有四个图使用的帧符号括起来的相互作用。使用框架支持重用的相互作用发生的相互作用

Unix

UNIX是一个计算机操作系统，能够在同一时间处理来自多个用户活动。Unix是在1969年在AT＆T贝尔实验室的Ken Thompson和Dennis Ritchie发起。本教程涉及内容：Unix,Unix教程,Unix操作系统,Unix命令等。

Unix 教程

UNIX是一个计算机操作系统，能够在同一时间处理来自多个用户活动。Unix是在1969年在AT＆T贝尔实验室的Ken Thompson和Dennis Ritchie发起。

读者

为初学者准备本教程，帮助他们了解基本Unix命令，UNIX shell脚本和各种实用工具，以先进的理念。

前提条件

我们假设你对操作系统和它的功能知之甚少。一个基本的了解各种不同的计算机概念，本教程给出的各种练习也将帮助你了解和入门Unix系统。

1 - UNIX快速参考指南

一个快速的Unix Unix系统用户参考指南。

UNIX快速参考指南

2 - UNIX工具和命令

在这里有一些有用的Unix工具和命令，UNIX系统工作需要非常频繁使用的。

UNIX工具和命令

3 - 有用的Unix资源

Unix 的网站，书籍和文章的集合在此页面。

- Bell Labs - UNIX操作系统的产生。简要介绍和UNIX操作系统的历史。
- BSD UNIX - FreeBSD是一种先进的现代的服务器，台式机和嵌入式计算机平台的UNIX操作系统。
- Linux Online - Linux是一套免费的类Unix作业系统，与世界各地的开发人员的协助下，最初由Linus Torvalds创建。
- Unix @ Wikipedia - UNIX操作系统的简要说明。
- The Unix Forums - 为Unix爱好者的论坛。与其他Unix专家分享您的意见和想法。

如果想在此页面列出您的网站，书籍或其他资源，欢迎联系 **yiibai.com#gmail.com**(用@代替**#**)

Unix是什么？ - Unix

Unix 是什么?

UNIX操作系统是一组程序，作为计算机和用户之间的链接。

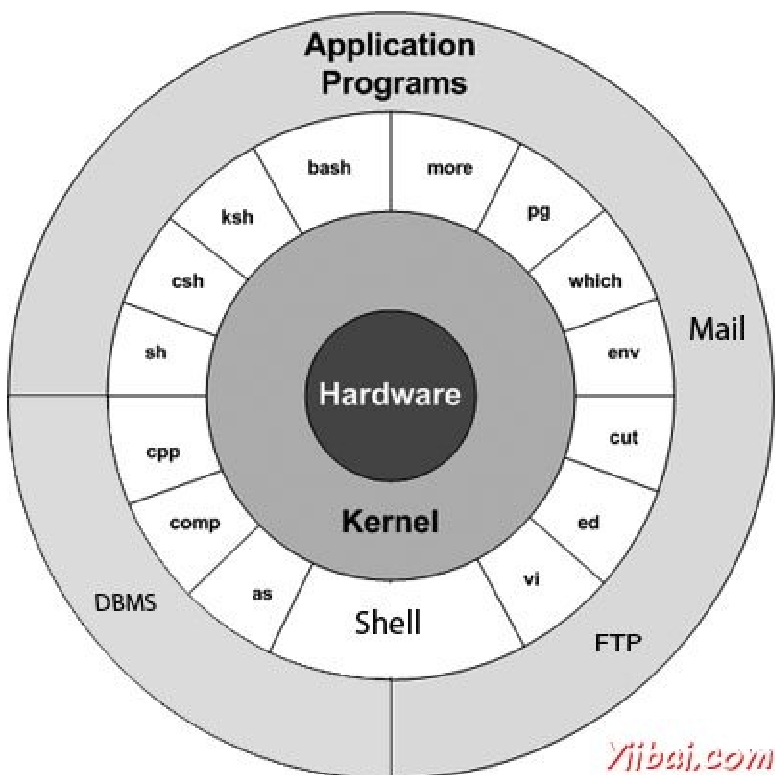
分配系统资源并协调所有的相关的详细信息的计算机的内部的计算机程序被调用的操作系统或内核。

用户与内核通信通过一个程序被称为shell。 shell是一个命令行解释器，它把由用户输入的命令，并将其转换成语言，理解由内核。

- Unix是最初开发于1969年，由一群在贝尔实验室，AT＆T员工，包括Ken Thompson, Dennis Ritchie, Douglas McIlroy, and Joe Ossanna.
- 可以在市场上有各种不同的Unix变种。Unix的Solaris，AIX，HP UNIX，BSD是几个例子。 Linux是Unix的一种，这是免费提供的。
- 有几个人可以同时使用UNIX计算机，因此，UNIX被称为一个多用户系统。
- 用户也可以在同一时间运行多个程序，因此UNIX被称为多任务。

UNIX架构：

下面是在UNIX系统的基本框图：



所有版本的UNIX的主要概念，是以下四个基础：

- 核心: 内核是操作系统的心脏。它与硬件和内存管理，任务调度和文件管理等任务。
- Shell: shell是实用工具，处理您的请求。当你在终端键入命令，shell解释命令和调用用的程序。shell采用标准语法的所有命令。C shell中Bourne shell和Korn外壳是最有名的shell，可与大多数的Unix变种。
- 命令和实用程序: 有各种不同的命令和实用程序，可以使用和活动。 cp, mv, cat 和 grep 等命令和实用程序的几个例子。有超过250个标准命令加上无数人通过第三方软件提供。所有的命令随之而来的各种可选方案。
- 文件和目录: UNIX中的所有数据被组织成文件。所有文件被组织成目录。这些目录被组织成一个树形结构，称为文件系统。

系统启动时：

如果有一台电脑，其中有UNIX操作系统上安装，那么只需要打开其电源，使用它。

只要打开电源，系统开始启动，最后它会提示登录到系统，这是一个活动登录到系统并使用它。

登陆 Unix:

当第一次连接到UNIX系统，通常会看到一个提示，如下面：

```
login:
```

要登录：

1. 有准备好用户ID和密码（用户识别）。请与系统管理员联系，如果还没有具备这些。
2. 在登录提示符下，键入您的用户ID，然后按ENTER键。userid是大小写敏感的，所以要确保键入它正是为系统管理员指派的。
3. 在密码提示符，键入密码，然后按ENTER键。密码是区分大小写。
4. 如果提供了正确的用户ID和密码，那么将被允许进入系统。如下的东西在屏幕上的信息和消息。

```
login : amrood
amrood's password:
Last login: Sun Jun 14 09:32:32 2009 from 62.61.164.73
$
```

它会提供一个命令提示符（有时称为$提示符），在那里可以输入一切命令。例如，检查日历，需要键入cal命令如下：

```
$ cal
     June 2009
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30

$
```

修改密码：

所有的Unix系统需要输入密码，以帮助确保文件和数据保持自己的系统本身是安全。下面是步骤更改密码：

1. 开始时，输入passwd命令提示，如下图所示。
2. 输入您目前正在使用的旧密码。
3. 输入您的新密码。始终保持密码足够复杂，因此，没有任何人可以猜测它。但要确保你记住它。
4. 您将需要再次输入验证密码。

```
$ passwd
Changing password for amrood
(current) Unix password:******
New UNIX password:*******
Retype new UNIX password:*******
passwd: all authentication tokens updated  successfully

$
```

注：显示星号（*），只是为了显示位置，需要输入当前密码和新密码，否则系统它不会告诉你，当键入任何字符。

目录和文件列表：

UNIX中的所有数据被组织成文件。所有文件被组织成目录。这些目录被组织成一个树形结构，称为文件系统。

可以使用ls命令列出所有的文件或目录，目录中可用。下面的例子使用ls命令使用-l选项。

```
$ ls -l
total 19621
drwxrwxr-x  2 amrood amrood      4096 Dec 25 09:59 uml
-rw-rw-r--  1 amrood amrood      5341 Dec 25 08:38 uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb 15  2006 univ
drwxr-xr-x  2 root   root        4096 Dec  9  2007 urlspedia
-rw-r--r--  1 root   root      276480 Dec  9  2007 urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov 25  2007 usr
-rwxr-xr-x  1 root   root        3192 Nov 25  2007 webthumb.php
-rw-rw-r--  1 amrood amrood     20480 Nov 25  2007 webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9  2007 yourfile.mid
-rw-rw-r--  1 amrood amrood    166255 Aug  9  2007 yourfile.swf

$
```

这里以d..... 开头的条目.....表示目录。例如UML，univ和girlspedia的其余的条目目录和文件。

你是谁？

当登录到系统，可能想知道： Who am I?

最简单的方法找出“who you are”，就是要进入whoami命令：

```
$ whoami
 amrood

$
```

在系统尝试。此命令列出当前登录的帐户名。可以试试，who am i 命令以及获得有关自己的信息。

谁在登录了？

有时你可能有兴趣知道是谁在同一时间也登录到计算机。

有三个命令可以，根据多少想了解其他用户： users, who 和 w.

```
$ users
 amrood bablu qadir

$ who
amrood ttyp0 Oct 8 14:10 (limbo)
bablu  ttyp2 Oct 4 09:08 (calliope)
qadir  ttyp4 Oct 8 12:09 (dent)

$
```

在系统上尝试w命令来检查输出。这将列出几个相关的信息系统中登录的用户。

注销：

当完成会话，你需要退出系统，以确保没有其他人访问您的文件，而伪装成你。

要注销：

1. 只需键入logout命令，在命令提示符下，该系统会清理一切并断开连接

系统关机：

最一致的方式通过命令行正确关闭一个Unix系统是使用下面的命令：

命令	描述
halt	Brings the system down immediately.
init 0	Powers off the system using predefined scripts to synchronize and clean up the system prior to shutdown
init 6	Reboots the system by shutting it down completely and then bringing it completely back up
poweroff	Shuts down the system by powering off.
reboot	Reboots the system.
shutdown	Shuts down the system.

你通常需要超级用户或根（在Unix系统中最有特权的帐户）关闭系统，但一些独立或个人拥有的Unix服务器，管理用户和普通用户有时可以这样做。

UNIX 文件管理 - Unix

UNIX中的所有数据被组织成文件。所有文件被组织成目录。这些目录被组织成一个树形结构，称为文件系统。

当使用UNIX工作以这种或那种方式，大部分的时间花在工作的文件。本教程将教你如何创建和删除文件，复制和重命名，创建链接到它们等。

在UNIX中，有三种基本类型的文件：

1. 普通文件: 普通文件在系统上的文件包含数据，文字，或程序指令。在本教程中，以操作普通的文件为例。
2. 目录: 目录特别及普通文件存储。对于用户熟悉Windows或Mac OS，UNIX目录相当于文件夹。
3. 特殊文件: 一些特殊的文件提供访问硬件，如硬盘，CD-ROM驱动器，调制解调器和以太网适配器。其他特殊文件是类似的别名或快捷键，使您能够访问一个单一的文件使用不同的名称。

文件列表：

要列出存储在当前目录中的文件和目录。使用下面的命令：

```
$ls
```

下面是上述命令的示例输出：

```
$ls

bin        hosts  lib     res.03
ch07       hw1    pub     test_results
ch07.bak   hw2    res.01  users
docs       hw3    res.02  work
```

ls命令支持，这将有助于获得更多的信息有关所列出的文件使用-1选项：

```
$ls -l
total 1962188

drwxrwxr-x  2 amrood amrood      4096 Dec 25 09:59 uml
-rw-rw-r--  1 amrood amrood      5341 Dec 25 08:38 uml.jpg
drwxr-xr-x  2 amrood amrood      4096 Feb 15  2006 univ
drwxr-xr-x  2 root   root        4096 Dec  9  2007 urlspedia
-rw-r--r--  1 root   root      276480 Dec  9  2007 urlspedia.tar
drwxr-xr-x  8 root   root        4096 Nov 25  2007 usr
drwxr-xr-x  2    200    300      4096 Nov 25  2007 webthumb-1.01
-rwxr-xr-x  1 root   root        3192 Nov 25  2007 webthumb.php
-rw-rw-r--  1 amrood amrood     20480 Nov 25  2007 webthumb.tar
-rw-rw-r--  1 amrood amrood      5654 Aug  9  2007 yourfile.mid
-rw-rw-r--  1 amrood amrood    166255 Aug  9  2007 yourfile.swf
drwxr-xr-x 11 amrood amrood      4096 May 29  2007 zlib-1.2.3
$
```

这里是所有列出的列信息：

1. 第一列：表示授予该文件的文件类型和权限。下面是描述所有类型的文件。
2. 第二列：代表所采取的内存块的文件或目录的数量。
3. 第三栏：表示文件所有者。是谁创建了这个文件的Unix用户。
4. 第四列：代表组的所有者。每个Unix用户将有关联的组。
5. 第五栏：表示文件大小（以字节为单位）。
6. 第六栏：表示当这个文件被创建或修改的最后一次的日期和时间。
7. 第七栏：表示文件名或目录名。

在 ls -l清单的例子，每一个文件的行开始d, -,或l。这些字符表示的文件类型列出。

Prefix	描述
-	Regular file, such as an ASCII text file, binary executable, or hard link.
b	Block special file. Block input/output device file such as a physical hard drive.
c	Character special file. Raw input/output device file such as a physical hard drive
d	Directory file that contains a listing of other files and directories.
l	Symbolic link file. Links on any regular file.
p	Named pipe. A mechanism for interprocess communications
s	Socket used for interprocess communication.

元字符：

在Unix元字符有特殊的含义。例如和？是元字符。我们使用匹配0个或多个字符，问号？匹配单个字符。

示例：

```
$ls ch*.doc
```

显示所有文件，其名称以ch开始和结束的。doc：

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc ch02-1.doc c
```

*元字符匹配任何字符。如果想显示所有文件只是文件结束，那么可以使用下面的命令：

```
$ls *.doc
```

隐藏文件：

一个无形的文件是一个点或者句号（.）的第一个字符是。 UNIX程序（包括shell）使用这些文件来存储配置信息。

隐藏文件的一些常见的例子包括以下文件：

- .profile: 是Bourne shell（sh）的初始化脚本
- .kshrc: Korn shell程序（KSH）初始化脚本
- .cshrc: C shell（csh）的初始化脚本
- .rhosts: 远程shell配置文件

要列出隐形文件，指定给ls-a选项：

```
$ ls -a

.         .profile       docs     lib     test_results
..        .rhosts        hosts    pub     users
.emacs    bin            hw1      res.01  work
.exrc     ch07           hw2      res.02
.kshrc    ch07.bak       hw3      res.03
$
```

- 单一点．：这表示当前目录。
- 双点 ..：这表示父目录。

注：我已经把星号（*），只是为了显示位置，需要输入当前密码和新密码，否则系统，它不会告诉你，当键入任何字符。

创建文件：

可以使用vi编辑器来创建任何类Unix系统上的普通文件。只需给下面的命令：

```
$ vi filename
```

上面的命令，将打开一个给定文件名的文件。会需要按键来进入编辑模式。一旦在编辑模式下，就可以开始写内容在文件，如下：

```
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
```

一旦完成，请执行以下步骤：

- 按Esc键出来的编辑模式。
- 按两个键Shift+ ZZ一起完全退出来的文件。

现在，就可以把filemame创建的文件在当前目录中。

```
$ vi filename
$
```

编辑文件：

使用vi编辑器，可以编辑现有的文件。我们将覆盖在一个单独的教程中详细。但总之，可以打开现有的文件如下：

```
$ vi filename
```

一旦文件被打开，在编辑模式下，可以使用 i 按键，然后，可以编辑文件。如果想在这里和那里的文件内，那么首先需要走出来的编辑模式下按ESC键，那么可以使用下面的键，将里面的一个文件：

- l 键移动到右侧。
- h 键移动到左侧。
- k 键向上移动在文件中。
- j 键向下移动一边在文件中。

因此，使用上面的键就可以将光标定位在任何想要的编辑。然后在编辑模式下，可以使用i键。编辑该文件，一旦完成后，按ESC和最后两个键Shift+ ZZ一起退出的文件。

显示文件的内容：

可以使用cat命令来查看一个文件的内容。下面是一个简单的例子，看看上面创建的文件的内容：

```
$ cat filename
This is unix file....I created it for the first time.....
I'm going to save this content in this file.
$
```

可以通过使用-b选项一起cat命令如下显示行号：

```
$ cat filename -b
1   This is unix file....I created it for the first time.....
2   I'm going to save this content in this file.
$
```

计算在一个文件中的单词：

可以使用wc命令行，字，字符包含在一个文件总数计数。以下是一个简单的例子来看看上面创建的文件的信息：

```
$ wc filename
2  19 103 filename
$
```

这里是所有的四列的细节：

1. 第一栏：表示文件中的行的总数。
2. 第二栏：表示文件中的总字数。
3. 第三栏：表示文件中的字节总数。这是实际的文件大小。
4. 第四栏：表示文件名。

在同一时间，以获得有关这些文件的信息，可以给多个文件。下面是简单的语法：

```
$ wc filename1 filename2 filename3
```

复制文件：

要复制的文件，请使用cp命令。该命令的基本语法是：

```
$ cp source_file destination_file
```

下面的例子，现有的文件的文件名创建一个副本。

```
$ cp filename copyfile
$
```

现在，发现多了一个文件在当前目录copyfile 。该文件将作为原始文件的filename完全相同。

重命名文件：

要更改名称的文件，使用mv命令。它的基本语法是：

```
$ mv old_file new_file
```

下面的例子将重命名现有文件 filename 修改为 newfile :

```
$ mv filename newfile
$
```

mv命令将现有的文件移动到新的文件完全。因此在这种情况下，在当前目录中有 newfile。

删除文件：

要删除现有的文件使用rm命令。它的基本语法是：

```
$ rm filename
```

注意：这可能是危险的，删除一个文件，因为它可能包含有用的信息。所以，要小心使用此命令时。建议使用-i选项的rm命令一起使用。

下面的例子将完全删除现有文件的文件名 filename：

```
$ rm filename
$
```

可以删除多个文件如下：

```
$ rm filename1 filename2 filename3
$
```

标准的UNIX流：

在正常情况下，每个Unix程序有三个流（文件）打开它时，它启动：

1. stdin : 这被称为作为标准输入和与其相关的文件的描述符是0。这也表示标准输入。UNIX程序会从STDIN读取默认的输入。
2. stdout : 这被称为作为标准输出和1相关的文件描述符。这也表示STDOUT。 UNIX程序会写入默认输出STDOUT
3. stderr : 这是被称为标准错误和2相关的文件描述符。这也代表STDERR。 UNIX程序会写在STDERR所有的错误消息。

UNIX 目录管理 - Unix

目录是一个文件，其唯一的工作就是存储文件的名称和相关信息。无论是普通的，特殊的或目录，包含所有文件，在目录中。

UNIX采用了分层结构组织的文件和目录。这种结构通常称为作为一个目录树。树有单个根节点，斜杠字符（/），和所有其他目录下面都包含它。

主目录：

目录中，你会发现自己，当第一次登录时被称为你的home目录。

你的工作一般会在你的home目录，创建组织文件的目录和子目录。

可以去home目录中随时使用下面的命令：

```
$cd ~
$
```

这里~表示主目录。如果想要去的任何其他用户的主目录中，然后使用下面的命令：

```
$cd ~username
$
```

最后一个目录，可以使用以下命令：

```
$cd -
$
```

绝对/相对路径名：

目录都被排列在顶部的根（/）在一个层次。在层次结构中的位置的任何文件描述由它的路径名。

用/分隔路径名的元素。路径名是绝对的，如果它被描述与根系，所以绝对路径总是以一个/.

这都是一些例子，绝对文件名.

```
/etc/passwd
/users/sjones/chem/notes
/dev/rdsk/Os3
```

也可以是相对当前工作目录的路径名。永远不会开始以/相对路径名。相对到用户amrood'的主目录，有些路径名可能看起来像这样：

```
chem/notes
personal/res
```

为了确定您所处的文件系统的层次结构内的任何时间，输入命令pwd打印当前工作目录：

```
$pwd
/user0/home/amrood

$
```

列出目录：

要列出目录中的文件，可以使用下面的语法：

```
$ls dirname
```

下面的例子列出包含的所有文件在/usr/local目录：

```
$ls /usr/local

X11       bin          gimp      jikes       sbin
ace       doc          include   lib         share
atalk     etc          info      man         ami
```

创建目录：

通过以下命令创建目录：

```
$mkdir dirname
```

在这里，目录是要创建的目录的绝对或相对路径名。例如，下面的命令：

```
$mkdir mydir
$
```

在当前目录下创建目录mydir。这里是另一个例子：

```
$mkdir /tmp/test-dir
$
```

此命令在/ tmp目录下创建目录test目录。 mkdir命令不产生任何输出，如果它成功地创建请求的目录。

如果在命令行上给多目录，mkdir 会创建的每个目录。例如：

```
$mkdir docs pub
$
```

在当前目录下创建目录 docs 和 pub。

创建父目录：

有时，当想创建一个目录，它的父目录或目录可能不存在。在这种情况下mkdir 会发出错误消息，如下所示：

```
$mkdir /tmp/amrood/test
mkdir: Failed to make directory "/tmp/amrood/test";
No such file or directory
$
```

在这种情况下，您可以指定命令mkdir-p选项。它会创建所有必需的目录。例如：

```
$mkdir -p /tmp/amrood/test
$
```

上面的命令创建所有必需的父目录。

删除目录：

目录可以使用rmdir命令删除如下：

```
$rmdir dirname
$
```

注意：要删除一个目录，请确保它是空的，这意味着不应该有这个目录里面的任何文件或子目录。

可以创建多个目录的时间如下：

```
$rmdir dirname1 dirname2 dirname3
$
```

上述命令将删除的目录dirname1 dirname2，dirname2如果它们是空的。 rmdir命令不产生任何输出，如果它是成功的。

改变目录：

可以使用cd命令做多变化的主目录：可以用它来指定一个有效的绝对或相对路径切换到任意目录。语法如下：

```
$cd dirname
$
```

在这里，dirname是目录的名称。例如，下面的命令：

```
$cd /usr/local/bin
$
```

更改到目录 /usr/local/bin。在这个目录中，你可以cd到目录 /usr/home/amrood 使用下面的相对路径：

```
$cd ../../home/amrood
$
```

重命名目录：

mv (move) 命令也可以用来重命名一个目录。语法如下：

```
$mv olddir newdir
$
```

可以重命名目录mydir 为 yourdir，如下：

```
$mv mydir yourdir
$
```

目录 .（点）和 ..（点点）

文件名 .（点）代表当前的工作目录和文件名 ..（点点）代表当前工作目录的上一级目录，通常称为父目录。

如果我们输入命令来显示当前工作目录的文件清单，并使用-a选项列出的所有文件和-l选项提供长列出，这是结果。

```
$ls -la
drwxrwxr-x    4    teacher   class   2048  Jul 16 17.56 .
drwxr-xr-x    60   root              1536  Jul 13 14:18 ..
----------    1    teacher   class   4210  May 1 08:27 .profile
-rwxr-xr-x    1    teacher   class   1948  May 12 13:42 memo
$
```

UNIX 文件权限/访问模式 - Unix

文件所有权是UNIX的一个重要组成部分，提供了一个安全的方法，用于存储文件。 UNIX中的每个文件具有以下属性：

- 所有者权限: 所有者的权限，确定所采取的操作文件的拥有者可以执行文件。
- 组权限: 该组的权限确定哪些操作用户，谁是一个文件所属的组的成员，可以执行该文件。
- 其它权限: 别人的权限表明什么么样的操作，所有其他用户可以执行该文件。

权限指示：

当用ls-l命令显示各种信息相关的文件权限如下：

```
$ls -l /home/amrood
-rwxr-xr--  1 amrood   users 1024  Nov 2 00:10  myfile
drwxr-xr--- 1 amrood   users 1024  Nov 2 00:10  mydir
```

这里第一列代表不同的访问模式，即关联的权限的文件或目录。

权限被分成三组，每个组中的位置表示一个特定的权限，这个顺序：读（r），写（w），执行（x）：

- 前三个字符（2-4）表示文件所有者的权限。例如 -rwxr-xr-- 代表读（r），写（w）和执行（x）许可。
- 第二组的三个字符（5-7）由文件所属组的权限。例如-rwxr-xr--代表该组读（r）和执行（x）权限，但没有写权限。
- 最后一组的三个字符（8-10）表示其他的权限。例如rwxr-XR - 代表其他只允许读（r）

文件访问模式：

一个文件的权限是一个Unix系统安全防御的第一线。 Unix权限的基本构建块的读，写和执行权限，说明如下：

1. Read:

即阅读，查看该文件的内容。

2. Write:

修改或删除的文件的内容。

3. Execute:

作为一个程序执行权限的用户可以运行一个文件。

目录访问方式：

目录访问方式列出，像其他文件中相同的方式组织。有一些差异，需要提及：

1. Read:

到一个目录中的访问意味着用户可以读取内容。用户可以看看目录里面的文件名。

2. Write:

访问装置，用户可以添加或删除的文件的目录的内容。

3. Execute:

执行目录并没有真正有很大的意义，认为这是一个遍历权限。

用户必须拥有的bin目录的执行权限，以执行ls或cd命令。

更改权限：

要改变文件或目录的权限，使用chmod（其他模式）命令。有两种方法使用chmod：符号模式和绝对模式。

使用chmod符号模式：

对于一个初学者，修改文件或目录的权限最简单的方法是使用符号模式。具有象征性的权限，可以添加，删除，或使用下表中的操作符，想要指定的权限集。

Chmod 操作符	描述
+	Adds the designated permission(s) to a file or directory.
-	Removes the designated permission(s) from a file or directory.
=	Sets the designated permission(s).

下面是一个例子，使用testfile的。运行ls -1 在 testfile将显示文件的权限如下：

```
$ls -l testfile
-rwxrwxr--  1 amrood   users 1024  Nov 2 00:10  testfile
```

然后每个例子chmod命令从前面的表上运行，其次testfile将ls-l，可以看到权限更改：

```
$chmod o+wx testfile
$ls -l testfile
-rwxrwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod u-x testfile
$ls -l testfile
-rw-rwxrwx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod g=r-x testfile
$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

这里是如何将这些命令放在一行：

```
$chmod o+wx,u-x,g=r-x testfile
$ls -l testfile
-rw-r-xrwx  1 amrood   users 1024  Nov 2 00:10  testfile
```

使用chmod绝对权限：

使用chmod命令修改权限的第二种方法是使用一些指定每个组的文件的权限。

每个权限分配一个值，按照下表中所示，每一组的总的权限提供了许多用于该集合。

Number	八进制权限表示	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

下面是一个例子，使用testfile。运行ls -1 在 testfile 将显示文件的权限如下：

```
$ls -l testfile
-rwxrwxr--  1 amrood   users 1024  Nov 2 00:10  testfile
```

然后每个例子chmod命令从前面的表上运行，其次testfile将ls-l，所以 可以看到权限更改：

```
$ chmod 755 testfile
$ls -l testfile
-rwxr-xr-x  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod 743 testfile
$ls -l testfile
-rwxr---wx  1 amrood   users 1024  Nov 2 00:10  testfile
$chmod 043 testfile
$ls -l testfile
----r---wx  1 amrood   users 1024  Nov 2 00:10  testfile
```

更改所有者和组：

在Unix上创建一个帐户，一个所有者分配给每个用户的ID和组ID。上面提到的所有的权限也被分配基础上的所有者和组。

两个命令都可以改变文件的所有者和组：

1. chown: chown命令代表“更改所有者”，是用来改变文件所有者。
2. chgrp: chgrp命令代表“更改组”，是用来改变文件的组。

所有权变更：

chown命令更改文件的所有权。基本语法如下：

```
$ chown user filelist
```

用户的值可以是一个用户在系统上的一个用户在系统上的用户ID（UID）的名称。

下面的例子：

```
$ chown amrood testfile
$
```

更改给定的文件的所有者用户amrood。

注意：超级用户root拥有不受限制的能力改变任何文件的所有权，但普通用户只能改变他们所拥有的文件的所有者。

更改组所有权：

chrgp命令组文件的所有权更改。基本语法如下：

```
$ chgrp group filelist
```

组的值可以是系统或组ID（GID）系统上的一组一组的名称。

下面的例子：

```
$ chgrp special testfile
$
```

组给定的文件更改特殊组。

SUID和SGID文件权限：

通常，当执行一个命令，它会以完成其任务，必须具有特殊权限的执行。

作为一个例子，当用passwd命令更改您的密码，新密码存储在文件 /etc/shadow文件。

作为一个普通用户，没有读或写访问此文件出于安全原因，但是当改变你的密码，需要写这个文件的权限。这意味着passwd程序给额外的权限，这样就可以写入文件/ etc / shadow中。

其他权限的程序通过设置用户ID（SUID）和设置组ID（SGID）位被称为一种机制。

当执行一个程序，启用了SUID位，继承该程序的所有者的权限。启动程序的用户的权限运行程序没有设置SUID位。

这才是真正为SGID。通常情况下，节目组权限执行，而是您的组将被改变只是对这一计划的程序的组所有者。

如果权限SUID和SGID位会出现字母“s”。位于所有者的权限位执行权限通常会位于SUID位将被“S”。例如，命令

```
$ ls -l /usr/bin/passwd
-r-sr-xr-x  1   root   bin  19031 Feb 7 13:47  /usr/bin/passwd*
$
```

这表明了SUID位设置该命令是由root。大写字母S在执行位置，而不是一个小写字母s表示，没有设置执行位。

如果粘位上启用的目录，文件只能被删除，如果是以下用户：

- 所有者粘性目录
- 所有者被删除的文件
- 超级用户root

要设置SUID和SGID位上的任何目录尝试以下操作：

```
$ chmod ug+s dirname
$ ls -l
drwsr-sr-x 2 root root  4096 Jun 19 06:45 dirname
$
```

UNIX 环境 - Unix

Unix的一个重要概念是环境，被定义的环境变量。有些系统通过环境变量，还有一些由shell，或任何程序加载另一个程序。

变量是一个字符串，我们分配一个值。分配的值可以是一个数字，文本，文件名，移动设备，或任何其他类型的数据。

例如，首先我们设定一个变量测试，然后我们使用echo命令来访问它的值：

```
$TEST="Unix Programming"
$echo $TEST
Unix Programming
```

需要注意的是环境变量设置，而无需使用$符号，但访问它们时，我们使用$符号作为前缀。这些变量保持它们的值，直到我们设计出来shell。

当你登录到系统中，shell经过一个阶段称为初始化设置各种环境。这通常是一个两步的过程，涉及的的shell阅读下列文件：

- /etc/profile
- profile

过程如下：

1. shell进行检查，看是否存在文件 /etc/profile文件。
2. 如果它存在，当shell读取。否则，此文件将被跳过。不显示错误消息。
3. shell检查，看看是否该文件。配置文件存在于你的home目录。主目录是开始在登录后的目录
4. 如果它存在，当shell读取它，否则shell跳过。不显示错误消息。

只要这两个文件被读取，shell将显示一个提示：

```
$
```

这是提示这里你可以输入命令，以让他们执行。

注 - 这里详述shell初始化过程适用于所有的Bourne型shell，但所使用的是bash和ksh一些额外的文件。

.profile 文件:

文件/etc/profile文件是由UNIX机器的系统管理员，包含shell初始化所需的信息系统上的所有用户。

该文件 .profile是在你的控制之下。您可以添加尽可能多的 shell定制信息，只要想这个文件。最小信息集，需要配置包括：

- 使用的终端的类型
- 在其中定位命令的目录的列表
- 列表变量，影响终端的外观和风格。

可以检查 .profile，在home目录。使用vi编辑器打开它，并检查所有的变量设置环境。

设置终端类型：

通常情况下，使用的终端自动配置通过登录或getty程序。有时，自动配置过程中猜测终端不正确。

如果终端设置不正确，命令的输出可能看起来很奇怪，或者可能无法正确与shell交互。

为了确保，这是没有的情况下，大多数用户他们的终端设置到最低共同如下：

```
$TERM=vt100
$
```

设置路径：

当输入任何命令在命令提示符下，shell也有定位才可以执行的命令。

PATH变量中指定位置的的shell看起来应该命令。一般它被设置如下：

```
$PATH=/bin:/usr/bin
$
```

这里每一个由冒号分开的各个条目，目录。如果要求的shell来执行命令并不能找到它在PATH变量中的任何目录中，出现类似下面的消息：

```
$hello
hello: not found
$
```

有变量，如PS1和PS2在下一节讨论。

PS1和PS2变量：

命令提示符下的shell显示为字符存储在变量PS1。可以改变这个变量是你想要的任何东西。只要改变它，它会被用来由shell，从这一点上。

例如，如果发出命令：

```
$PS1='=>'
=>
=>
=>
```

将成为提示=>。要设置PS1的价值，因此，它显示的工作目录，发出以下命令：

```
=>PS1="[u@h w]$"
[root@ip-72-167-112-17 /var/www/yiibai/unix]$
[root@ip-72-167-112-17 /var/www/yiibai/unix]$
```

此命令的结果是，提示显示用户的用户名，机器名（hostname），工作目录。

有相当多的PS1的值参数可以用来作为转义序列，试图限制了，提示没有太多的信息。

Escape Sequence	描述
Current time, expressed as HH:MM:SS.	
d	Current date, expressed as Weekday Month Date Newline.
s	Current shell environment.
W	Working directory.
w	Full path of the working directory.
u	Current user.s username.
h	Hostname of the current machine.
#	Command number of the current command. Increases with each new command entered.
$	If the effective UID is 0 (that is, if you are logged in as root), end the prompt with the # character; otherwise, use the $.

可以使自己的变化，每次登录时，或者可以将它添加到您的配置文件中所做的更改会自动在PS1。

当你发出一个命令，是不完整的，shell将显示辅助提示，等待完成该命令，然后再次按Enter键。

默认的次级提示>（大于号），但可以改变重新定义PS2 shell变量：

下面的例子使用默认的次级提示：

```
$ echo "this is a
> test"
this is a
test
$
```

下面的例子重新定义PS2定制提示：

```
$ PS2="secondary prompt->"
$ echo "this is a
secondary prompt->test"
this is a
test
$
```

环境变量：

以下是部分重要的环境变量列表。如上所述，这些变量将被设置和访问：

变量	描述
DISPLAY	Contains the identifier for the display that X11 programs should use by default.
HOME	Indicates the home directory of the current user: the default argument for the cd built-in command.
IFS	Indicates the Internal Field Separator that is used by the parser for word splitting after expansion.
LANG	LANG expands to the default system locale; LC_ALL can be used to override this. For example, if its value is pt_BR, then the language is set to (Brazilian) Portuguese and the locale to Brazil.
LD_LIBRARY_PATH	On many Unix systems with a dynamic linker, contains a colon-separated list of directories that the dynamic linker should search for shared objects when building a process image after exec, before searching in any other directories.
PATH	Indicates search path for commands. It is a colon-separated list of directories in which the shell looks for commands.
PWD	Indicates the current working directory as set by the cd command.
RANDOM	Generates a random integer between 0 and 32,767 each time it is referenced.
SHLVL	Increments by one each time an instance of bash is started. This variable is useful for determining whether the built-in exit command ends the current session.
TERM	Refers to the display type
TZ	Refers to Time zone. It can take values like GMT, AST, etc.
UID	Expands to the numeric user ID of the current user, initialized at shell startup.

以下是几个环境变量的样本示例：

```
$ echo $HOME
/root
]$ echo $DISPLAY

$ echo $TERM
xterm
$ echo $PATH
/usr/local/bin:/bin:/usr/bin:/home/amrood/bin:/usr/local/bin
$
```

Unix 基本工具(打印，电子邮件) - Unix

到目前为止，你必须有一些想法关于Unix操作系统和性质，其基本的命令。本教程将涵盖一些非常基本的，但重要的Unix工具，可以使用在你的工作中。

打印文件：

在UNIX系统上打印文件之前，你可能需要对其进行格式化，调整页边距，突出一些的话，等。大多数文件也可以被打印而无需重新格式化，但原始的打印输出可能不会看起来相当不错。

许多UNIX版本包括两个功能强大的文本格式化，nroff和troff。他们不包括在本教程中。

pr 命令:

pr命令做轻微的格式在终端屏幕上的文件或打印机。例如，如果你在一个文件中的名称有一个长长的清单，你可以在屏幕上格式化成两个或更多列。

下面是pr命令的语法：

```
pr option(s) filename(s)
```

pr改变格式的文件，只在屏幕上或打印的副本，它并不修改原文件。下表列出了一些pr选项：

Option	描述
-k	Produces k columns of output
-d	Double-spaces the output (not on all pr versions).
-h "header"	Takes the next item as a report header.
-t	Eliminates printing of header and top/bottom margins.
-l PAGE_LENGTH	Set the page length to PAGE_LENGTH (66) lines. Default number of lines of text 56.
-o MARGIN	Offset each line with MARGIN (zero) spaces.
-w PAGE_WIDTH	Set page width to PAGE_WIDTH (72) characters for multiple text-column output only.

使用pr之前，这里是一个样本文件的内容，名为“ food

```
$cat food
Sweet Tooth
Bangkok Wok
Mandalay
Afghani Cuisine
Isle of Java
Big Apple Deli
Sushi and Sashimi
Tio Pepe's Peppers
........
$
```

让我们使用pr命令头*Restaurants*做一个两列的报告：

```
$pr -2 -h "Restaurants" food
Nov  7  9:58 1997  Restaurants   Page 1

Sweet Tooth              Isle of Java
Bangkok Wok              Big Apple Deli
Mandalay                 Sushi and Sashimi
Afghani Cuisine          Tio Pepe's Peppers
........
$
```

lp和lpr命令：

LP或lpr命令打印文件的纸张上，而不是在屏幕上显示。一旦准备好使用pr命令的格式，可以使用这些命令与您的电脑连接的打印机上打印文件。

您的系统管理员可能已经在您的站点设置一个默认打印机。在默认打印机上打印文件命名的food ，使用lp或lpr命令，如在这个例子：

```
$lp food
request id is laserp-525  (1 file)
$
```

lp命令显示了一个ID，你可以用它来取消打印作业或检查其状态。

- 如果您正在使用lp命令，可以使用-N Num 选项，打印的份数Num 。随着lpr命令，可以使用-Num 相同。
- 如果有多个与共享的网络连接的打印机，那么你可以选择打印机lp命令，为了同样的目的，可以使用-P打印机选项随着lpr命令使用-D打印机选项。这里的打印机是打印机的名称。

lpstat和lpq指令：

lpstat命令显示打印机队列中的请求ID，所有者，文件大小，发送打印工作时，请求的状态。

使用lpstat-o如果想看到所有的输出要求，而不是只是自己。请求他们将印刷的顺序：

```
$lpstat -o
laserp-573  john  128865  Nov 7  11:27  on laserp
laserp-574  grace  82744  Nov 7  11:28
laserp-575  john   23347  Nov 7  11:35
$
```

使用lpq给出了略有不同的信息比用lpstat - o：

```
$lpq
laserp is ready and printing
Rank   Owner      Job  Files                 Total Size
active john       573  report.ps             128865 bytes
1st    grace      574  ch03.ps ch04.ps       82744 bytes
2nd    john       575  standard input        23347 bytes
$
```

在这里，第一行显示打印机状态。如果打印机被禁用或缺纸，你可能会看到不同的消息，在此第一行。

cancel 和 lprm 命令:

取消终止lp命令打印请求。 lprm命令终止的lpr请求。您可以指定请求的ID（LP或LPQ显示）或打印机的名称。

```
$cancel laserp-575
request "laserp-575" cancelled
$
```

要取消当前正在打印什么样的请求，无论其ID，只需输入取消和打印机名称：

```
$cancel laserp
request "laserp-573" cancelled
$
```

lprm命令命令将取消积极的工作，如果它属于你。否则，你可以给工作数字作为参数，或使用破折号（ - ），删除所有作业：

```
$lprm 575
dfA575diamond dequeued
cfA575diamond dequeued
$
```

lprm命令将告诉实际的文件名从打印机队列中删除。

发送Email:

您可以使用Unix的邮件命令的发送和接收邮件。下面是的语法发送电子邮件：

```
$mail [-s subject] [-c cc-addr] [-b bcc-addr] to-addr
```

这里是重要的mail 命令相关的选项：

Option	描述
-s	Specify subject on command line.
-c	Send carbon copies to list of users. List should be a comma-separated list of names.
-b	Send blind carbon copies to list. List should be a comma-separated list of names.

以下的例子来发送一条测试消息admin@yahoo.com。

```
$mail -s "Test Message" admin@yahoo.com
```

预期然后键入消息，其次是 "control-D" 开头的行。要停止只需键入点（.）如下：

```
Hi,

This is a test
.
Cc:
```

您可以将一个完整的文件使用重定向<操作如下：

```
$mail -s "Report 05/06/07" admin@yahoo.com < demo.txt
```

检查传入邮件，在UNIX系统中，只需键入电子邮件如下：

UNIX 管道和过滤器 - Unix

您可以连接在一起，使两个命令从一个程序的输出成为下一个程序的输入。以这种方式连接的两个或多个命令可以形成一个管。

要制作一个管道，放了两个命令在命令行之间的竖线（|）。

当一个程序将其从另一个程序的输入，该输入执行某些操作，并写入到标准输出的结果，它被称为作为一个过滤器。

grep命令：

grep程序搜索一个或多个文件的行有一定的模式。语法是：

```
$grep pattern file(s)
```

这个名字源于“grep”的ED（UNIX行编辑器）命令g/re/p指“全局搜索所有包含一个正则表达式和打印。”

正则表达式是一些纯文本（一个字，例如）和/或特殊字符用于模式匹配。

grep最简单的用法是看一个字组成的一个模式。它可以用在管道中，因而只有那些行包含一个给定的字符串的输入文件被发送到标准输出。如果你不给grep的读取一个文件名时，它读取标准输入的所有过滤方案的工作方式：

```
$ls -l | grep "Aug"
-rw-rw-rw-   1 john  doc     11008 Aug  6 14:10 ch02
-rw-rw-rw-   1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-r--   1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-r--   1 carol doc      1605 Aug 23 07:35 macros
$
```

有各种不同的选项，可以一起使用grep命令：

Option	描述
-v	Print all lines that do not match pattern.
-n	Print the matched line and its line number.
-l	Print only the names of files with matching lines (letter "l")
-c	Print only the count of matching lines.
-i	Match either upper- or lowercase.

接下来，让我们使用一个正则表达式，告诉grep来查找线“carol”，由零个或多个其他字符在正则表达式简称为“*”），再其次是“Aug”。

在这里，我们使用-i选项来区分大小写的搜索：

```
$ls -l | grep -i "carol.*aug"
-rw-rw-r--   1 carol doc      1605 Aug 23 07:35 macros
$
```

sort命令：

sort命令安排行的文字，字母或数字。下面的例子各种行在food文件：

```
$sort food
Afghani Cuisine
Bangkok Wok
Big Apple Deli
Isle of Java
Mandalay
Sushi and Sashimi
Sweet Tooth
Tio Pepe's Peppers
$
```

sort命令默认情况下，按字母顺序排列的文本行。有很多的选择，控制排序：

Option	描述
-n	Sort numerically (example: 10 will sort after 2), ignore blanks and tabs.
-r	Reverse the order of sort.
-f	Sort upper- and lowercase together.
+x	Ignore first x fields when sorting.

超过两个命令可以链接到一个管道。以先前的管道使用grep的例子，我们可以进一步在August 修改的文件大小顺序进行排序。

下面的管道由命令ls，grep和排序：

```
$ls -l | grep "Aug" | sort +4n
-rw-rw-r--  1 carol doc      1605 Aug 23 07:35 macros
-rw-rw-r--  1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-rw-  1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-rw-  1 john  doc     11008 Aug  6 14:10 ch02
$
```

August 大小顺序修改这条管道在你的目录中的所有文件进行排序，并将它们打印到终端屏幕。行数字顺序排序选项+4Ñ跳过四个字段（字段由空格分隔），然后排序。

pg 和更多的命令：

一个长的输出通常会压缩在屏幕上，但如果你运行文本通过或pg作为一个过滤器，每屏显示的文本后，显示停止。

让我们假设你有一个长的目录列表。为了使其更易于阅读排序列表，通过管道输出如下：

```
$ls -l | grep "Aug" | sort +4n | more
-rw-rw-r--  1 carol doc      1605 Aug 23 07:35 macros
-rw-rw-r--  1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-rw-  1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-r--  1 john  doc     14827 Aug  9 12:40 ch03
    .
    .
    .
-rw-rw-rw-  1 john  doc     16867 Aug  6 15:56 ch05
--More--(74%)
```

屏幕将填补文本，其中包括按文件大小顺序排序的行了一屏。在屏幕底部是更迅速，在那里你可以通过排序的文本键入一个命令来移动。

当你完成这个屏幕上，可以使用任何的讨论更多程序中列出的命令。

UNIX 进程管理 - Unix

当你执行一个程序在UNIX系统上，该系统为该程序创建一个特殊的环境。这个环境包含系统运行的程序，如果没有其他程序在系统上运行所需要的一切。

每当你发出命令在UNIX中，创建或启动一个新的进程。当你尝试了合列出目录的内容，你启动了一个进程。一个过程，简单来说，就是一个正在运行的程序的一个实例。

操作系统将跟踪进程称为PID或进程ID通过五位ID号。该系统中的每个进程都有一个唯一的pid。

PIDS最终重复，因为所有可能的数字和未来的PID滚动或重新开始。在任何一个时间，没有两个具有相同的pid的进程在系统中存在，因为它是UNIX使用跟踪每个进程的pid。

启动进程：

当你启动一个进程（运行命令），您可以运行它有两种方法：

- 前台进程
- 后台进程

前台进程：

默认情况下，每一个过程，你开始在前台运行。从键盘输入，并将其输出到屏幕上。

使用ls命令，你可以看到这一点。如果我要列出当前目录中的所有文件，就可以使用下面的命令：

```
$ls ch*.doc
```

这将显示所有文件的名称开始ch和结束以 .doc：

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

程序在前台运行，输出定向到我的屏幕，如果ls命令希望任何输入（事实并非如此），它会等待从键盘。

当程序在前台运行，并采取太多的时间，我们不能运行任何其他命令（启动任何其他的进程），因为提示将无法使用，直到程序完成其加工出来。

后台进程：

后台进程运行，而无需连接到你的键盘。如果后台进程需要任何键盘输入，它会等待。

进程在后台运行的优点是，你可以运行其他命令，你不必等待，直到它完成启动另一个！

启动一个后台进程的最简单的方法是在命令末尾添加一个符号（&）。

```
$ls ch*.doc &
```

这也将显示所有文件的名称开始ch和结束的 .doc：

```
ch01-1.doc   ch010.doc  ch02.doc    ch03-2.doc
ch04-1.doc   ch040.doc  ch05.doc    ch06-2.doc
ch01-2.doc   ch02-1.doc
```

这里如果ls命令希望任何输入（事实并非如此），进入停止状态，直到我把它移动到前台，并给它的数据从键盘。

也就是说第一行包含有关后台进程的信息 - 工作的数量和进程ID。你需要了解作业的数量，操纵它在前台和后之间。

如果你现在按下回车键，看到以下内容：

```
[1]   +   Done                 ls ch*.doc &
$
```

第一行告诉你ls命令的后台进程成功完成。第二个是另一个命令提示。

列出正在运行的进程：

这是很容易看到自己的进程运行的ps（进程状态）命令如下：

```
$ps
PID       TTY      TIME        CMD
18358     ttyp3    00:00:00    sh
18361     ttyp3    00:01:31    abiword
18789     ttyp3    00:00:00    ps
```

其中最常用的标志 **-f** (f for full) 选项，它提供了更多的信息，如下面的示例所示：

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

这里是ps-f命令显示的所有字段的描述：

Column	描述
UID	User ID that this process belongs to (the person running it).
PID	Process ID.
PPID	Parent process ID (the ID of the process that started it).
C	CPU utilization of process.
STIME	Process start time.
TTY	Terminal type associated with the process
TIME	CPU time taken by the process.
CMD	The command that started this process.

还有其他的选择可以一起使用ps命令：

Option	描述
-a	Shows information about all users
-x	Shows information about processes without terminals.
-u	Shows additional information like -f option.
-e	Display extended information.

停止进程：

几种不同的方法可以做到在结束过程。通常情况下，从一个基于控制台的命令，发送CTRL + C按键（默认中断字符）将退出命令。这工作进程正在运行时在前台模式。

如果一个进程在后台模式下运行，那么首先你需要得到其作业ID，使用ps命令后，你可以使用kill命令来杀死进程如下：

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

这里kill命令将终止first_one的过程。如果一个进程忽略了正规的kill命令，你可以使用kill-9进程ID如下：

```
$kill -9 6738
Terminated
```

父进程和子进程：

每个UNIX进程有两个ID号分配给它的进程ID（PID）和父进程ID（PPID）。系统中的每一个用户进程都有一个父进程。

您运行的大部分命令作为其父的shell。检查ps -f 这个命令列出进程ID和父进程ID。

僵尸和孤儿进程：

通常情况下，当一个孩子被杀害，被告知父进程通过一个SIGCHLD信号。然后，父进程可以做一些其他的任务，或者需要重新启动一个新的子进程。然而，有时父进程在子进程之前被杀死。在这种情况下，“父的所有进程，”init进程，成为新的PPID（父进程ID）。有时，这些进程被称为孤儿进程。

当一个进程被杀死，ps列出可能仍然显示一个Z状态的过程。这是一具僵尸或解散过程。进程是死的，不被使用。这些过程是不同于孤立进程。它们是已完成执行的处理，但仍然有进程表中的一个条目。

守护进程：

守护进程是系统相关的后台进程，往往根和服务请求来自其他进程的权限运行。

守护进程没有控制终端。它不能打开/dev/tty。如果你用“ps-ef”，并期待在tty字段中，所有的守护进程将有一个吗？ tty。

更清楚的是，仅仅是一个守护进程在后台运行的进程，通常在等待一些事情发生，它是有能力的工作，就像一台打印机守护进程正在等待打印命令。

如果你有一个程序需要做长期处理，那么使其成为一个守护进程，它在后台运行。

top命令：

top命令是一个非常有用的工具，用于快速显示按各种标准排序的的进程。

这是一个互动的诊断工具，经常更新，并显示物理和虚拟内存，CPU使用率，平均负载，以及您繁忙的的进程的信息。

下面是简单的语法来运行top命令，并看到不同进程的CPU利用率的统计：

```
$top
```

作业ID与进程ID：

背景和暂停的的进程通常是通过操纵作业号（作业ID）。这个数字是不同的进程ID的使用，因为它是短暂的。

此外，作业可包含串联或在同一时间运行，并联的多个进程，因此，使用的作业ID是容易跟踪单个进程。

UNIX 网络实用工具 - Unix

当你的工作需要在分布式环境中的沟通与远程用户，还需要访问远程Unix机器。

有一些Unix实用程序，这是特别有用的计算在一个网络的分布式环境中的用户。本教程列出几个：

Ping实用程序：

ping命令发送一个回送请求到主机在网络上可用。使用这个命令你可以检查如果您的远程主机或不响应。

ping命令是有用的项目如下：

- 跟踪和隔离硬件和软件的问题。
- 确定网络和各种外部主机的状态。
- 测试，测量和管理网络。

语法

以下是简单的语法使用ping命令：

```
$ping hostname or ip-address
```

上面的命令将开始打印后每一秒的响应。要退出来命令可以终止按CNTRL+ C键。

例子：

以下是查询，马上在网络上可用的主机的例子：

```
$ping google.com
PING google.com (74.125.67.100) 56(84) bytes of data.
64 bytes from 74.125.67.100: icmp_seq=1 ttl=54 time=39.4 ms
64 bytes from 74.125.67.100: icmp_seq=2 ttl=54 time=39.9 ms
64 bytes from 74.125.67.100: icmp_seq=3 ttl=54 time=39.3 ms
64 bytes from 74.125.67.100: icmp_seq=4 ttl=54 time=39.1 ms
64 bytes from 74.125.67.100: icmp_seq=5 ttl=54 time=38.8 ms
--- google.com ping statistics ---
22 packets transmitted, 22 received, 0% packet loss, time 21017ms
rtt min/avg/max/mdev = 38.867/39.334/39.900/0.396 ms
$
```

如果主机不存在，那么它的行为会像这样：

```
$ping giiiiiigle.com
ping: unknown host giiiiigle.com
$
```

FTP工具：

这里，FTP代表文件传输协议。该工具可以帮助您上传和下载你的文件从一台计算机到另一台计算机。

FTP工具有其自己的一套UNIX命令一样，让你可以执行任务，如：

- 连接并登录到远程主机。
- 导航目录。
- 列出目录内容
- 上传和下载文件
- ASCII，EBCDIC或二进制传输文件

语法

以下是简单的语法使用ping命令：

```
$ftp hostname or ip-address
```

上面的命令会提示你输入登录ID和密码。一旦你验证，你将有机会获得登录帐户的主目录，你就可以执行各种命令。

下面列出了几个有用的命令：

命令	描述
put filename	Upload filename from local machine to remote machine.
get filename	Download filename from remote machine to local machine.
mput file list	Upload more than one files from local machine to remove machine.
mget file list	Download more than one files from remote machine to local machine.
prompt off	Turns prompt off, by default you would be prompted to upload or download movies using mput or mget commands.
prompt on	Turns prompt on.
dir	List all the files available in the current directory of remote machine.
cd dirname	Change directory to dirname on remote machine.
lcd dirname	Change directory to dirname on local machine.
quit	Logout from the current login.

应当指出，所有的文件将被下载或上传或从当前目录。如果你想在一个特定的目录下上传您的文件，那么你首先切换到该目录所需的文件，然后上传。

例子：

下面的例子显示几个命令：

```
$ftp amrood.com
Connected to amrood.com.
220 amrood.com FTP server (Ver 4.9 Thu Sep 2 20:35:07 CDT 2009)
Name (amrood.com:amrood): amrood
331 Password required for amrood.
Password:
230 User amrood logged in.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls.
total 1464
drwxr-sr-x   3 amrood   group       1024 Mar 11 20:04 Mail
drwxr-sr-x   2 amrood   group       1536 Mar  3 18:07 Misc
drwxr-sr-x   5 amrood   group        512 Dec  7 10:59 OldStuff
drwxr-sr-x   2 amrood   group       1024 Mar 11 15:24 bin
drwxr-sr-x   5 amrood   group       3072 Mar 13 16:10 mpl
-rw-r--r--   1 amrood   group     209671 Mar 15 10:57 myfile.out
drwxr-sr-x   3 amrood   group        512 Jan  5 13:32 public
drwxr-sr-x   3 amrood   group        512 Feb 10 10:17 pvm3
226 Transfer complete.
ftp> cd mpl
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening data connection for /bin/ls.
total 7320
-rw-r--r--   1 amrood   group       1630 Aug  8 1994  dboard.f
-rw-r-----   1 amrood   group       4340 Jul 17 1994  vttest.c
-rwxr-xr-x   1 amrood   group     525574 Feb 15 11:52 wave_shift
-rw-r--r--   1 amrood   group       1648 Aug  5 1994  wide.list
-rwxr-xr-x   1 amrood   group       4019 Feb 14 16:26 fix.c
226 Transfer complete.
ftp> get wave_shift
200 PORT command successful.
150 Opening data connection for wave_shift (525574 bytes).
226 Transfer complete.
528454 bytes received in 1.296 seconds (398.1 Kbytes/s)
ftp> quit
221 Goodbye.
$
```

telnet实用程序：

很多时候，你需要远程连接到这台机器上远程的Unix机器和工作。 Telnet是一种实用工具，允许用户在一个站点的计算机进行连接，登录，然后在计算机上进行工作，在另一个站点。

一旦你使用telnet登录，您可以执行远程连接的机器上所有活动。这里是例如telnet会话：

```
C:>telnet amrood.com
Trying...
Connected to amrood.com.
Escape character is '^]'.

login: amrood
amrood's Password:
*****************************************************
*                                                   *
*                                                   *
*    WELCOME TO AMROOD.COM                          *
*                                                   *
*                                                   *
*****************************************************

Last unsuccessful login: Fri Mar  3 12:01:09 IST 2009
Last login: Wed Mar  8 18:33:27 IST 2009 on pts/10

   {  do your work }

$ logout
Connection closed.
C:>
```

finger 实用工具：

finger命令显示给定主机上的用户信息。可以是本地或远程主机。

出于安全原因，手指可能会被禁止在其他系统上。

以下是简单的语法使用finger命令：

检查所有本地机器上登录的用户如下：

```
$ finger
Login     Name       Tty      Idle  Login Time   Office
amrood               pts/0          Jun 25 08:03 (62.61.164.115)
```

获取本地机器上的一个特定的用户信息：

```
$ finger amrood
Login: amrood                           Name: (null)
Directory: /home/amrood                 Shell: /bin/bash
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115
No mail.
No Plan.
```

检查所有用户在远程机器上登录的情况如下：

```
$ finger @avtar.com
Login     Name       Tty      Idle  Login Time   Office
amrood               pts/0          Jun 25 08:03 (62.61.164.115)
```

获得远程机器上的一个特定的用户信息：

```
$ finger amrood@avtar.com
Login: amrood                           Name: (null)
Directory: /home/amrood                 Shell: /bin/bash
On since Thu Jun 25 08:03 (MST) on pts/0 from 62.61.164.115
No mail.
No Plan.
```

vi编辑器教程 - Unix

有很多种Unix和编辑文件对我来说最好的方法之一是使用面向屏幕的文本编辑器vi。这个编辑器让您编辑在上下文中的其他文件中的行线。

现在你会发现这就是所谓的VIM vi编辑器的改进版本。这里VIM代表ViIM被证明。

被普遍认为是在VI编辑器，因为在Unix事实上的标准：

- 它通常可在Unix系统中的使用。
- 它的实现是非常的全面。
- 它需要很少的资源。
- 这是比任何其他的编辑器，如ed或前更加用户友好。

可以使用vi编辑器来编辑现有的文件或创建一个新的文件从头。您也可以使用这个编辑器只读取一个文本文件。

启动vi编辑器：

有以下方式，你就可以开始使用vi编辑器：

命令	描述
vi filename	Creates a new file if it already does not exist, otherwise opens existing file.
vi -R filename	Opens an existing file in read only mode.
view filename	Opens an existing file in read only mode.

testfile将创建一个新的文件，如果它已经不存在于当前的工作目录下面的例子：

```
$vi testfile
```

因此，你会看到类似如下的画面：

```
|
~
~
~
~
~
~
~
~
~
~
~
~
"testfile" [New File]
```

光标之后你会发现在每行的符号（~）。一个波浪线表示未使用的线路。如果行不以波浪号开始，似乎是空白，有空格，制表符，换行符，或其他一些存在nonviewable字符。

所以，现在你已经打开了一个文件开始。在继续之前，让我们了解一些微小但重要的概念解释如下。

操作模式：

用vi编辑器工作时，你会遇到以下两种模式：

1. 命令模式：此模式可让您执行管理任务，如保存文件，执行命令，将光标移动，切割和粘贴线或字（唬弄），并寻找和更换。在这种模式下，无论你是作为一个命令解释。
2. 插入模式：此模式使您可以插入到该文件中的文本。一切的在此模式下输入被解释为输入，最后，它被放置在该文件。

vi的总是开始在命令模式。要输入文字，您必须在插入模式。来插入模式，你只需键入i。要插入模式，按Esc键，就会把你背到命令模式。

提示：如果你不知道你是在哪个模式，按Esc键两次，那么你就可以在命令模式。您使用vi编辑器打开一个文件，并开始键入一些字符，然后在命令模式来理解上的差异。

退出**vi**：

退出vi的命令是 :q。一旦在命令模式下，键入冒号，和“q”，依次是返回。如果您的文件已经以任何方式修改，编辑器会提醒你这一点，而不是让你戒烟。要忽略此消息，不保存退出vi的命令是:q! 这可以让你退出VI不保存任何的变化。

保存编辑器内容的命令是 :w。您可以结合上面的命令quit命令，或者 :wq和返回。

保存更改并退出vi的最简单的方法是ZZ命令。当你在命令模式下:wq， 它会做相当于输入ZZ。

您可以指定一个不同的文件名保存到指定的名称：W之后。例如，如果你希望你工作，另一名为文件名的文件名保存该文件，您可以键入:w filename2中和返回。尝试一次。

移动在一个文件中：

要左右移动在一个文件中，而不会影响你的文字，您必须在命令模式下（按Esc键两次）。这里有一些你可以用它来走动一次一个字符的命令：

命令	描述
k	Moves the cursor up one line.
j	Moves the cursor down one line.
h	Moves the cursor to the left one character position.
l	Moves the cursor to the right one character position.

有以下两个重要点要注意：

- vi是大小写敏感的，所以你需要使用命令时，要特别注意大小写。
- 大多数命令在vi中可以行动发生的时候，你想由数量开头。例如，2j移动光标两条线下来的光标位置。

还有很多其他的方法来移动在vi一个文件内。请记住，你必须在命令模式（按Esc键两次）。这里更多一些命令，你可以用它来走动文件：

命令	描述
0 or |	Positions cursor at beginning of line.
$	Positions cursor at end of line.
w	Positions cursor to the next word.
b	Positions cursor to previous word.
(	Positions cursor to beginning of current sentence.
)	Positions cursor to beginning of next sentence.
E	Move to the end of Blank delimited word
{	Move a paragraph back
}	Move a paragraph forward
[[	Move a section back
]]	Move a section forward
n|	Moves to the column n in the current line
1G	Move to the first line of the file
G	Move to the last line of the file
nG	Move to nth line of the file
:n	Move to nth line of the file
fc	Move forward to c
Fc	Move back to c
H	Move to top of screen
nH	Moves to nth line from the top of the screen
M	Move to middle of screen
L	Move to botton of screen
nL	Moves to nth line from the bottom of the screen
:x	Colon followed by a number would position the cursor on line number represented by **x**

控制命令：

使用命令后，您可以使用控制键：

命令	描述
CTRL+d	Move forward 1/2 screen
CTRL+d	Move forward 1/2 screen
CTRL+f	Move forward one full screen
CTRL+u	Move backward 1/2 screen
CTRL+b	Move backward one full screen
CTRL+e	Moves screen up one line
CTRL+y	Moves screen down one line
CTRL+u	Moves screen up 1/2 page
CTRL+d	Moves screen down 1/2 page
CTRL+b	Moves screen up one page
CTRL+f	Moves screen down one page
CTRL+I	Redraws screen

编辑文件：

要编辑的文件，你需要在插入模式。有很多方法进入插入模式，命令模式：

Command	描述
i	Inserts text before current cursor location.
I	Inserts text at beginning of current line.
a	Inserts text after current cursor location.
A	Inserts text at end of current line.
o	Creates a new line for text entry below cursor location.
O	Creates a new line for text entry above cursor location.

删除字符：

这里是清单可用于在一个打开的文件中删除字符和线条的重要的命令：

Command	描述
x	Deletes the character under the cursor location.
X	Deletes the character before the cursor location.
dw	Deletes from the current cursor location to the next word.
d^	Deletes from current cursor position to the beginning of the line.
d$	Deletes from current cursor position to the end of the line.
D	Deletes from the cursor position to the end of the current line.
dd	Deletes the line the cursor is on.

正如上面所提到的，大多数命令可以在vi行动发生的时候，你想由数量开头。例如，2个删除光标下的位置和2dd删除光标在两行两个字符。

我会强烈建议所有上面的命令，然后再继续正常练习。

更改命令：

你也可以改变在vi中的字符，单词或行而不删除他们。以下是相关的命令：

Command	描述
cc	Removes contents of the line, leaving you in insert mode.
cw	Changes the word the cursor is on from the cursor to the lowercase w end of the word.
r	Replaces the character under the cursor. vi returns to command mode after the replacement is entered.
R	Overwrites multiple characters beginning with the character currently under the cursor. You must use **Esc** to stop the overwriting.
s	Replaces the current character with the character you type. Afterward, you are left in insert mode.
S	Deletes the line the cursor is on and replaces with new text. After the new text is entered, vi remains in insert mode.

“复制”和“粘贴”命令：

您可以复制线或从一个地方的话，那么你就可以过去，他们在另一个地方使用下面的命令：

Command	描述
yy	Copies the current line.
yw	Copies the current word from the character the lowercase w cursor is on until the end of the word.
p	Puts the copied text after the cursor.
P	Puts the yanked text before the cursor.

高级的命令：

有一些先进的简化现在的编辑器，以便更有效地利用vi的命令：

Command	描述
J	Join the current line with the next one. A count joins that many lines.
<<	Shifts the current line to the left by one shift width.
>>	Shifts the current line to the right by one shift width.
~	Switch the case of the character under the cursor.
^G	Press CNTRL and G keys at the same time to show the current filename and the status.
U	Restore the current line to the state it was in before the cursor entered the line.
u	Undo the last change to the file. Typing 'u' again will re-do the change.
J	Join the current line with the next one. A count joins that many lines.
:f	Displays current position in the file in % and file name, total number of file.
:f filename	Renames current file to filename.
:w filename	Write to file filename.
:e filename	Opens another file with filename.
:cd dirname	Changes current working directory to dirname.
:e #	Use to toggle between two opened files.
:n	In case you open multiple files using vi, use :n to go to next file in the series.
:p	In case you open multiple files using vi, use :p to go to previous file in the series.
:N	In case you open multiple files using vi, use :N to go to previous file in the series.
:r file	Reads file and inserts it after current line
:nr file	Reads file and inserts it after line n.

Word和字符搜索：

vi编辑器有两种类型的搜索字符串和字符。对于一个字符串搜索，/和？命令的使用。当您启动这些命令，刚刚键入的命令将显示在底线上，在那里你输入特定的字符串来寻找。

这两个命令的不同在搜索方向替换：

- /命令向前搜索（向下）在文件中。
- ？命令向后搜索（向上）在文件中。

n和N命令重复以前的搜索命令，分别在相同或相反的方向。有些字符具有特殊的含义，同时使用搜索命令和前面的搜索表达式的一部分被列为一个反斜杠（）。

Character	描述
^	Search at the beginning of the line. (Use at the beginning of a search expression.)
.	Matches a single character.
*****	Matches zero or more of the previous character.
$	End of the line (Use at the end of the search expression.)
[	Starts a set of matching, or non-matching expressions.
<	Put in an expression escaped with the backslash to find the ending or beginning of a word.
>	See the '<' character description above.

同一行内的字符搜索搜索命令后，寻找输入的字符。f和F命令仅在当前行上的字符搜索。 f 向前搜索和F搜索向后光标移动到的位置找到的字符。

t和T命令搜索仅在当前行上的字符，但对于T，光标移动到该字符前的位置，和T行的字符之后的位置向后搜索。

set命令：

你可以改变它的外表和感觉使用以下屏VI：设置命令。要使用这些命令，你必须在命令模式下，然后键入 :set其次任何下列选项：

命令	描述
:set ic	Ignores case when searching
:set ai	Sets autoindent
:set noai	To unset autoindent.
:set nu	Displays lines with line numbers on the left side.
:set sw	Sets the width of a software tabstop. For example you would set a shift width of 4 with this command: **:set sw=4**
:set ws	If *wrapscan* is set, if the word is not found at the bottom of the file, it will try to search for it at the beginning.
:set wm	If this option has a value greater than zero, the editor will automatically "word wrap". For example, to set the wrap margin to two characters, you would type this: **:set wm=2**
:set ro	Changes file type to "read only"
:set term	Prints terminal type
:set bf	Discards control characters from input

运行命令：

vi有能力在编辑器中运行命令。要运行一个命令，你只需要进入命令模式输入 :! 命令。

例如，如果你要检查文件是否存在，然后再尝试保存您的文件，文件名，你可以键入:! ls在屏幕上，你会看到ls和ls的输出。

当你按任意键（或命令的转义序列），您将返回到您的vi会话。

替换文本：

替换命令(:s/)，使您能够快速替换单词或组的话，在你的文件。下面是一个简单的语法：

```
:s/search/replace/g
```

g代表全局。这个命令的结果是，所有出现光标的上线改变。

重要提示：

这里用vi你成功的关键点是：

- 您必须在命令模式下使用命令。 （按Esc键两次在任何时候，以确保您在命令模式中。）
- 你必须小心正确使用所有命令。
- 您必须在插入模式下输入文字。

Unix 正则表达式SED - Unix

正则表达式是一个字符串，它可以用来描述几个字符序列。使用正则表达式是由几个不同的Unix命令，包括 ed, sed, awk, grep，并且，在较为有限的程度上扩展 vi.

本教程将教你如何使用正则表达式使用 sed.

这里流编辑器sed的代表是面向流的编辑器，它是专门用于执行脚本创建。因此，所有的输入送入通过到stdout，它不会改变输入文件。

调用 sed:

在我们开始之前，让我们确保你有一个本地副本 /etc/passwd 文件的文本文件，用sed。

正如前面提到的，可以调用sed的发送数据通过管道如下：

```
$ cat /etc/passwd | sed
Usage: sed [OPTION]... {script-other-script} [input-file]...

  -n, --quiet, --silent
                 suppress automatic printing of pattern space
  -e script, --expression=script
...............................
```

cat命令转储 /etc/passwd文件的内容通过管道进入sed 模式空间sed 。是内部工作模式空间缓冲区，sed使用做其工作。

sed 一般语法：

以下是 sed 的一般语法

```
/pattern/action
```

在这里，模式是一个正则表达式，动作是下表中给出的命令之一。如果省略模式，执行操作的每一行，正如我们上面看到的。

斜线字符（/），环绕模式是必需的，因为它们被用来作为分隔符。

Range	描述
p	Prints the line
d	Deletes the line
s/pattern1/pattern2/	Substitutes the first occurrence of pattern1 with pattern2.

用sed删除所有行：

再次调用sed，但这个时候告诉sed使用编辑命令删除行，由单字母d表示：

```
$ cat /etc/passwd | sed 'd'
$
```

调用sed 发送文件，通过管道，而是可以指示sed来读取数据文件，在下面的例子。

下面的命令做完全一样的东西，以前的尝试，没有 cat 命令：

```
$ sed -e 'd' /etc/passwd
$
```

sed 位址：

SED还了解到一种叫做地址。位址是特定的地点，在一个文件或一个特定的编辑命令应适用范围。当sed遇到没有地址，在该文件中的每一行上执行其操作。

以下命令将sed 命令你已经使用了一个基本的地址：

```
$ cat /etc/passwd | sed '1d' |more
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
$
```

请注意，数字1之前添加删除编辑命令。这告诉sed执行编辑命令的第一行上的文件。在这个例子中，sed将删除第一行 /etc/password，并打印文件的其余部分。

sed 地址范围：

所以如果你想从文件中删除多个行？用sed，您可以指定一个地址范围如下：

```
$ cat /etc/passwd | sed '1, 5d' |more
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
$
```

上面的命令将开始从1至5的所有行。所以，删除前五行。

试试下面的地址范围：

Range	描述
'4,10d'	Lines starting from 4th till 10th are deleted
'10,4d'	Only 10th line is deleted, because sed does not work in reverse direction.
'4,+5d'	This will match line 4 in the file, delete that line, continue to delete the next five lines, and then cease its deletion and print the rest
'2,5!d'	This will deleted everything except starting from 2nd till 5th line.
'1~3d'	This deletes the first line, steps over the next three lines, and then deletes the fourth line. Sed continues applying this pattern until the end of the file.
'2~2d'	This tells sed to delete the second line, step over the next line, delete the next line, and repeat until the end of the file is reached.
'4,10p'	Lines starting from 4th till 10th are printed
'4,d'	This would generate syntax error.
',10d'	This would also generate syntax error.

注：使用p动作时，你应该使用-n选项，以避免重复行式打印。检查以下两条命令之间的区别：

```
$ cat /etc/passwd | sed -n '1,3p'
```

检查上面的命令没有-n作为如下：

```
$ cat /etc/passwd | sed '1,3p'
```

替换命令：

替换命令，用s表示，将您指定的其他任何字符串中指定的任何字符串代替。

用一个字符串代替另一个，你需要有一些方式告诉sed，你的第一个字符串结束，并开始替换字符串。这是传统上是由两个字符串bookending斜线（/）字符。

首次出现一行字符串根字符串amrood与下面的命令替代。

```
$ cat /etc/passwd | sed 's/root/amrood/'
amrood:x:0:0:root user:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
..........................
```

这是非常重要的，需要注意的是替代sed的只有第一次出现的行上。如果字符串根不止一次发生在一行的第一个匹配项将被替换。

告诉sed执行全局替换，添加字母g结束的命令如下：

```
$ cat /etc/passwd | sed 's/root/amrood/g'
amrood:x:0:0:amrood user:/amrood:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
...........................
```

替代标志：

还有一些其他有用的g标志除了可以传递的标志，你可以一次指定多个。

标志	描述
g	Replace all matches, not just the first match.
NUMBER	Replace only NUMBERth match.
p	If substitution was made, print pattern space.
w FILENAME	If substitution was made, write result to FILENAME.
I or i	Match in a case-insensitive manner.
M or m	In addition to the normal behavior of the special regular expression characters ^ and $, this flag causes ^ to match the empty string after a newline and $ to match the empty string before a newline.

使用替代字符串分隔符：

您可能会发现自己不得不做一个替换在一个字符串，其中包含斜线字符。在这种情况下，您可以指定不同的分隔，提供指定的字符后的s。

```
$ cat /etc/passwd | sed 's:/root:/amrood:g'
amrood:x:0:0:amrood user:/amrood:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

在上面的例子中，我们使用：作为分隔符，而不是斜线（/），因为我们试图搜索/root，而不是简单的root。

替换空字符：

使用空替换字符串从 /etc/passwd 文件中完全删除root字符串：

```
$ cat /etc/passwd | sed 's/root//g'
:x:0:0::/:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
```

地址替换：

如果你想用quiet 在第10行字符串替换字符串的sh，您可以指定如下：

```
$ cat /etc/passwd | sed '10s/sh/quiet/g'
root:x:0:0:root user:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/quiet
```

同样，做一个地址范围替换，你可以做类似以下内容：

```
$ cat /etc/passwd | sed '1,5s/sh/quiet/g'
root:x:0:0:root user:/root:/bin/quiet
daemon:x:1:1:daemon:/usr/sbin:/bin/quiet
bin:x:2:2:bin:/bin:/bin/quiet
sys:x:3:3:sys:/dev:/bin/quiet
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

正如你可以看到从输出前五行字符串的sh改变quiet，但其余各行均保持不变。

匹配的命令：

你会使用-n选项一起使用p选项打印所有匹配的行，如下所示：

```
$ cat testing | sed -n '/root/p'
root:x:0:0:root user:/root:/bin/sh
[root@ip-72-167-112-17 amrood]# vi testing
root:x:0:0:root user:/root:/bin/sh
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

使用正则表达式：

在匹配模式中，你可以使用正则表达式，它提供了更多的灵活性。

检查下面的例子匹配所有的行开始守护进程，然后删除它们：

```
$ cat testing | sed '/^daemon/d'
root:x:0:0:root user:/root:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
```

下面的例子将删除所有的行以sh结束：

```
$ cat testing | sed '/sh$/d'
sync:x:4:65534:sync:/bin:/bin/sync
```

下表列出了四个特殊字符在正则表达式中是非常有用的。

字符	描述
^	Matches the beginning of lines.
$	Matches the end of lines.
.	Matches any single character.
*	Matches zero or more occurrences of the previous character
[chars]	Matches any one of the characters given in chars, where chars is a sequence of characters. You can use the - character to indicate a range of characters.

匹配字符：

看几个表达式元字符演示使用。例如，下面的模式：

表达式	描述
/a.c/	Matches lines that contain strings such as a+c, a-c, abc, match, and a3c, whereas the pattern
/a*c/	Matches the same strings along with strings such as ace, yacc, and arctic.
/[tT]he/	Matches the string The and the:
/^$/	Matches Blank lines
/^.*$/	Matches an entire line whatever it is.
/ */	Matches one or more spaces
/^$/	Matches Blank lines

下表列出了一些常用的字符集：

Set	描述
[a-z]	Matches a single lowercase letter
[A-Z]	Matches a single uppercase letter
[a-zA-Z]	Matches a single letter
[0-9]	Matches a single number
[a-zA-Z0-9]	Matches a single letter or number

字符类关键词：

一些特殊的关键字是常用的正则表达式，特别是GNU工具，采用正则表达式。这些sed的正则表达式是非常有用的，因为它们简化了的东西，增强可读性。

例如，字符a到z以及A到Z的字符构成的字符的其中一类，具有关键字 [[:alpha:]]

使用字母字符类的关键字，只有那些行在 /etc/syslog.conf 文件，一个字母开始，这个命令打印：

```
$ cat /etc/syslog.conf | sed -n '/^[[:alpha:]]/p'
authpriv.*                          /var/log/secure
mail.*                              -/var/log/maillog
cron.*                              /var/log/cron
uucp,news.crit                      /var/log/spooler
local7.*                            /var/log/boot.log
```

下表是GNU sed的可用字符类中的关键字的完整列表。

Character Class	描述
[[:alnum:]]	Alphanumeric [a-z A-Z 0-9]
[[:alpha:]]	Alphabetic [a-z A-Z]
[[:blank:]]	Blank characters (spaces or tabs)
[[:cntrl:]]	Control characters
[[:digit:]]	Numbers [0-9]
[[:graph:]]	Any visible characters (excludes whitespace)
[[:lower:]]	Lowercase letters [a-z]
[[:print:]]	Printable characters (noncontrol characters)
[[:punct:]]	Punctuation characters
[[:space:]]	Whitespace
[[:upper:]]	Uppercase letters [A-Z]
[[:xdigit:]]	Hex digits [0-9 a-f A-F]

与符号引用：

sed 字元代表的模式相匹配的内容。例如，假设你有一个文件名为phone.txt的完整电话号码，如下面的：

```
5555551212
5555551213
5555551214
6665551215
6665551216
7775551217
```

你想更容易阅读的括号包围的区域码（前三位）。要做到这一点，你可以使用符号替换字符，像这样：

```
$ sed -e 's/^[[:digit:]][[:digit:]][[:digit:]]/(&)/g' phone.txt
(555)5551212
(555)5551213
(555)5551214
(666)5551215
(666)5551216
(777)5551217
```

在模式匹配第3位，然后使用要更换这3个数字与周围的括号。

使用多个sed命令：

您可以使用多个sed命令在一个单一的sed命令如下：

```
$ sed -e 'command1' -e 'command2' ... -e 'commandN' files
```

这里命令通过commandN是前面讨论过的类型的sed命令。这些命令被施加到给定的文件的文件列表中的各行。

我们可以使用相同的机制，上面写的电话号码的例子如下：

```
$ sed -e 's/^[[:digit:]]{3}/(&)/g'
                      -e 's/)[[:digit:]]{3}/&-/g' phone.txt
(555)555-1212
(555)555-1213
(555)555-1214
(666)555-1215
(666)555-1216
(777)555-1217
```

注：在上面的例子中，而不是重复字符类关键字 [[:digit:]]三次，取而代之的是{3}，这意味着匹配前面的正则表达式三次。在这里，我用 断行运行此命令之前你应该删除。

返回参考：

符号元字符是有用的，但更为有用的是能够定义特定的区域，在一个正则表达式，这样你就可以替换字符串中引用它们。通过定义一个正则表达式的特定部分，你可以参考那些部分特别提到字符。

要做返回引用，你必须首先定义一个区域，然后参考该区域。要定义一个区域，你插入反斜杠括号，围绕感兴趣区域。环绕反斜杠第一区域，然后引用 1， 2 第二区域，依此类推。

假设phone.txt有以下文字：

```
(555)555-1212
(555)555-1213
(555)555-1214
(666)555-1215
(666)555-1216
(777)555-1217
```

现在尝试下面的命令：

```
$ cat phone.txt | sed 's/(.*))(.*-)(.*$)/Area
                       code: 1 Second: 2 Third: 3/'
Area code: (555) Second: 555- Third: 1212
Area code: (555) Second: 555- Third: 1213
Area code: (555) Second: 555- Third: 1214
Area code: (666) Second: 555- Third: 1215
Area code: (666) Second: 555- Third: 1216
Area code: (777) Second: 555- Third: 1217
```

注意：在上面的例子中括号内的每个正则表达式将引用1 2，依此类推。在这里，我用断行运行此命令之前你应该删除。

Unix 文件系统基础 - Unix

文件系统是一个分区或磁盘上的文件的逻辑集合。分区是信息的容器，如果需要的话，可以跨越整个硬盘驱动器。

你的硬盘驱动器，可以有不同的分区通常只包含一个文件系统，如一个文件系统 / 文件系统或其他包含/home文件系统。

每个分区的一个文件系统允许不同的文件系统的维护和管理逻辑。

在Unix中的一切都被认为是一个文件，包括物理设备，如DVD-ROM，USB设备，软盘驱动器，等等。

目录结构：

Unix 使用层次结构的文件系统的结构，很象一个倒置的树，在该文件系统的基础上，并从那里扩展的所有其他目录的根目录（/）。

UNIX 文件系统中的文件和目录的集合，具有以下属性：

- 它有一个根目录（/），其中包含其他文件和目录。
- 每个文件或目录被唯一标识，它的名字，它的目录，以及一个唯一的标识符，通常被称为一个inode。
- 按照惯例，根目录下有一个inode号为2和3的lost + found目录中有一个inode号。 inode编号0和1不使用。指定-i选项的ls命令可以看到文件的inode编号。
- 这是自包含的。没有一个文件系统，以及任何其他的之间的依赖关系。

目录有特定的目的，普遍持相同类型的信息，轻松定位文件。以下是主要的Unix版本上存在的目录：

目录	描述
/	This is the root directory which should contain only the directories needed at the top level of the file structure.
/bin	This is where the executable files are located. They are available to all user.
/dev	These are device drivers.
/etc	Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages.
/lib	Contains shared library files and sometimes other kernel-related files.
/boot	Contains files for booting the system.
/home	Contains the home directory for users and other accounts.
/mnt	Used to mount other temporary file systems, such as cdrom and floppy for the CD-ROM drive and floppy diskette drive, respectively
/proc	Contains all processes marked as a file by process number or other information that is dynamic to the system.
/tmp	Holds temporary files used between system boots
/usr	Used for miscellaneous purposes, or can be used by many users. Includes administrative commands, shared files, library files, and others
/var	Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
/sbin	Contains binary (executable) files, usually for system administration. For example*fdisk* and *ifconfig* utlities.
/kernel	Contains kernel files

浏览文件系统：

现在你了解基本的文件系统，你就可以开始导航到你需要的文件。以下是您将使用浏览系统的命令：

命令	描述
cat filename	Displays a filename.
cd dirname	Moves you to the directory identified.
cp file1 file2	Copies one file/directory to specified location.
file filename	Identifies the file type (binary, text, etc).
find filename dir	Finds a file/directory.
head filename	Shows the beginning of a file.
less filename	Browses through a file from end or beginning.
ls dirname	Shows the contents of the directory specified.
mkdir dirname	Creates the specified directory.
more filename	Browses through a file from beginning to end.
mv file1 file2	Moves the location of or renames a file/directory.
pwd	Shows the current directory the user is in.
rm filename	Removes a file.
rmdir dirname	Removes a directory.
tail filename	Shows the end of a file.
touch filename	Creates a blank file or modifies an existing file.s attributes.
whereis filename	Shows the location of a file.
which filename	Shows the location of a file if it is in your PATH.

您可以使用联机帮助帮助这里提到的每个命令的语法检查完成。

df 命令:

第一种方式来管理你的分区空间使用df（磁盘空闲）命令。命令df-K（可用磁盘）以KB为单位显示磁盘空间使用情况，如下图所示：

```
$df -k
Filesystem      1K-blocks      Used   Available Use% Mounted on
/dev/vzfs        10485760   7836644     2649116  75% /
/devices                0         0           0   0% /devices
$
```

一些目录，如 /devices，显示0字节，使用和可用列的能力，以及0%。这些特殊的文件系统（或虚拟），虽然它们驻留在磁盘上/下，它们本身不占用磁盘空间。

df -k 输出所有的Unix系统上大致相同。下面是它通常包括：

Column	描述
Filesystem	The physical file system name.
kbytes	Total kilobytes of space available on the storage medium.
used	Total kilobytes of space used (by files).
avail	Total kilobytes available for use.
capacity	Percentage of total space used by files.
Mounted on	What the file system is mounted on.

你可以使用-h（人类可读的）选项显示的输出格式，显示的大小更容易理解的符号。

du 命令:

使用du（磁盘使用率）命令使您能够在一个特定的目录指定目录显示磁盘空间使用情况。

此命令是有帮助的，如果你要确定一个特定的目录多大的空间。下面的命令将显示每个目录所消耗的块数。单块可能需要512字节或1千字节，这取决于你的系统。

```
$du /etc
10     /etc/cron.d
126    /etc/default
6      /etc/dfs
...
$
```

-h选项使输出更容易理解：

```
$du -h /etc
5k    /etc/cron.d
63k   /etc/default
3k    /etc/dfs
...
$
```

安装的文件系统：

必须安装的文件系统，以便使用该系统。要看到什么是目前在您的系统上安装（可以使用），使用这个命令：

```
$ mount
/dev/vzfs on / type reiserfs (rw,usrquota,grpquota)
proc on /proc type proc (rw,nodiratime)
devpts on /dev/pts type devpts (rw)
$
```

/ mnt目录下，Unix的惯例，是位于临时装片（如CD-ROM驱动器，远程网络驱动器，软盘驱动器）。如果您需要挂载文件系统，可以使用mount命令的语法如下：

```
mount -t file_system_type device_to_mount directory_to_mount_to
```

例如，如果你想要的CD-ROM安装到目录 /mnt/cdrom，例如，您可以键入：

```
$ mount -t iso9660 /dev/cdrom /mnt/cdrom
```

假定您的CD-ROM设备名为 /dev/cdrom命令要挂载到 /mnt/cdrom。更具体的信息或帮助信息在命令行中键入mount -h ，请参阅安装手册页。

安装后，您可以使用cd 命令导航最新的文件系统挂载点。

卸载文件系统：

从您的系统中卸载（删除）文件系统，使用umount命令识别挂载点或设备

例如，要卸载光驱，使用下面的命令：

```
$ umount /dev/cdrom
```

mount 命令，使您能够访问您的文件系统，但最现代的Unix系统，自动装载功能，使这个过程对用户不可见，无需干预。

用户和组配额：

用户和组配额提供的机制，可以限制特定组内的单个用户或所有用户使用的空间量由管理员定义的值。

配额围绕两个限制，允许用户采取一些行动，如果量的空间或磁盘块数开始超过管理员定义的限制：

- Soft Limit: 如果用户超过定义的限制，有一个宽限期，允许用户腾出一些空间。
- Hard Limit: 当达到硬限制，无论在宽限期，没有进一步的文件或块可以分配。

有一些命令来管理配额：

命令	描述
quota	Displays disk usage and limits for a user of group.
edquota	This is a quota editor. Users or Groups quota can be edited using this command.
quotacheck	Scan a filesystem for disk usage, create, check and repair quota files
setquota	This is also a command line quota editor.
quotaon	This announces to the system that disk quotas should be enabled on one or more filesystems.
quotaoff	This announces to the system that disk quotas should be disabled off one or more filesystems.
repquota	This prints a summary of the disc usage and quotas for the specified file systems

您可以使用 联机帮助帮助 提到的每个命令的语法检查完成。

UNIX 用户管理 - Unix

在Unix系统上的账户有三种类型：

1. Root 账号: 这也被称为超级用户，并有完整的和不受约束的控制系统。一个超级用户可以运行任何命令，没有任何限制。该用户应承担作为一个系统管理员。
2. System 账号: 系统帐户是那些需要特定系统组件，例如电子邮件帐户和sshd的账户的操作。这些账户通常需要在您的系统上的一些特定功能，任何修改系统可能会受到不好的影响。
3. User 账号: 用户帐户提供交互式访问系统的用户和用户组。一般使用者通常分配给这些帐户，通常有有限的访问关键系统文件和目录。

UNIX支持组帐户的概念逻辑分组多个账户。每个帐户将任何组帐户的一部分。 Unix群组中起着重要的作用，在处理文件的权限和流程管理。

管理用户和组：

有三个主要的用户管理文件：

1. /etc/passwd: 保持用户帐户和密码信息。这个文件包含了大多数的Unix系统上的账户信息。
2. /etc/shadow: 相应的帐户保存加密口令。并非所有的系统支持此文件。
3. /etc/group: 此文件包含每个帐户的组信息。
4. /etc/gshadow: 此文件包含安全组的帐户信息。

检查上述所有文件使用cat命令。

以下是大多数Unix系统上可用来创建和管理帐户和组的命令：

命令	描述
useradd	Adds accounts to the system.
usermod	Modifies account attributes.
userdel	Deletes accounts from the system.
groupadd	Adds groups to the system.
groupmod	Modifies group attributes.
groupdel	Removes groups from the system.

您可以使用联机帮助帮助这里提到的每个命令的语法检查完成。

创建一个组

您需要创建组，然后才能再创建任何帐户，系统中必须使用现有组。你将不得不在 /etc/groups文件中列出的所有组。

所有默认组将系统帐户的特定群体，它是不推荐使用普通帐户。所以语法来创建一个新的帐户：

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

下面是详细的参数：

选项	描述
-g GID	The numerical value of the group's ID.
-o	This option permits to add group with non-unique GID
-r	This flag instructs groupadd to add a system account
-f	This option causes to just exit with success status if the specified group already exists. With -g, if specified GID already exists, other (unique) GID is chosen
groupname	Actaul group name to be created.

如果你不指定任何参数，那么系统将使用默认值。

以下示例将创建开发组的默认值，这是非常可以接受的大多数管理员。

```
$ groupadd developers
```

修改组：

要修改组，使用groupmod语法：

```
$ groupmod -n new_modified_group_name old_group_name
```

要改变developers_2 组的名称到开发组，输入：

```
$ groupmod -n developer developer_2
```

这里显示如何改变GID为545：

```
$ groupmod -g 545 developer
```

删除组：

要删除现有的组，所有你需要的是一个命令groupdel命令和组名。要删除的 financial 组，该命令是：

```
$ groupdel developer
```

这将删除组，没有任何与该组相关的文件。这些文件是由他们的所有者仍然可以访问。

创建一个帐户

让我们来看看如何在你的Unix系统上创建一个新的帐户。以下是语法来创建用户帐户：

```
useradd -d homedir -g groupname -m -s shell -u userid accountname
```

下面是详细的参数：

Option	描述
-d homedir	Specifies home directory for the account.
-g groupname	Specifies a group account for this account.
-m	Creates the home directory if it doesn't exist.
-s shell	Specifies the default shell for this account.
-u userid	You can specify a user id for this account.
accountname	Actual account name to be created

如果你不指定任何参数，那么系统将使用默认值。useradd命令修改了 /etc/passwd, /etc/shadow, 和 /etc/group文件，并创建一个主目录。

下面的例子将创建一个帐户 mcmohd 其主目录设置到 /home/mcmohd 和开发组。该用户将有Korn Shell的分配给它。

```
$ useradd -d /home/mcmohd -g developers -s /bin/ksh mcmohd
```

发出上述命令前，请确保你已经有开发组使用groupadd的命令创建。

一旦创建一个帐户，你可以设置其密码，使用passwd命令如下：

```
$ passwd mcmohd20
Changing password for user mcmohd20.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

当你输入 passwd accountname，它给你提供的密码你是超级用户，否则，你就可以改变你的密码使用相同的命令，但没有指定帐户名选项来改变。

修改帐户：

通过usermod命令使您可以更改现有的帐户，在命令行。它使用相同的参数，useradd命令，加上-l参数，它允许您更改帐户名。

举例来说，，更改帐户名称 mcmohd 到 mcmohd20，并相应地改变主目录，你会需要发出以下命令：

```
$ usermod -d /home/mcmohd20 -m -l mcmohd mcmohd20
```

删除一个帐户：

userdel命令可以用来删除现有用户。这是一个非常危险的命令，如果不小心使用。

只有一个参数或选项可用于命令：.r，删除帐户的主目录和邮件文件。

例如，删除帐户mcmohd20，您将需要发出以下命令：

```
$ userdel -r mcmohd20
```

如果你想保持她的主目录备份的目的，省略-r选项。您可以删除的主目录，在以后的时间。

UNIX 系统性能 - Unix

本教程的目的是引进性能分析师提供的免费工具，来监视和管理UNIX系统的性能，以及如何诊断和修复性能问题在Unix环境提供指引。

UNIX具有以下主要资源类型，需要进行监测和调整：

- CPU
- 内存
- 磁盘空间
- 通信线路
- I/O 时间
- 网络时间
- 应用程序

性能组件：

有以下主要的五个组成部分的系统总时间的推移：

组件	描述
User state CPU	The actual amount of time the CPU spends running the users program in the user state. It includes time spent executing library calls, but does not include time spent in the kernel on its behalf.
System state CPU	This is the amount of time the CPU spends in the system state on behalf of this program. All I/O routines require kernel services. The programmer can affect this value by the use of blocking for I/O transfers.
I/O Time and Network Time	These are the amount of time spent moving data and servicing I/O requests
Virtual Memory Performance	This includes context switching and swapping.
Application Program	Time spent running other programs - when the system is not servicing this application because another application currently has the CPU.

性能工具：

UNIX提供了重要的工具来测量和微调Unix系统的性能：

命令	描述
nice/renice	Run a program with modified scheduling priority
netstat	Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships
time	Time a simple command or give resource usage
uptime	System Load Average
ps	Report a snapshot of the current processes.
vmstat	Report virtual memory statistics
gprof	Display call graph profile data
prof	Process Profiling
top	Display system tasks

您可以使用联机帮助帮助这里提到的每个命令的语法检查完成。

UNIX 系统日志 - Unix

Unix 系统有一个非常灵活和强大的日志系统，它可以让您记录几乎任何你能想象和操作日志，以获取所需要的信息。

许多版本 UNIX 提供了一个通用的日志工具，称为：syslog。个别程序需要有记录的信息发送到 syslog 信息。

Unix的系统日志主机配置，统一的系统日志记录工具。该系统采用一个集中的系统运行程序 /etc/syslogd 或 /etc/syslog 日志记录进程。.

系统日志的操作是相当简单的。程序日志条目发送到syslogd，参考的配置文件在/etc/syslogd.conf 或 /etc/syslog，找到一个匹配时，所需的日志文件写入日志消息。

有四个基本的 syslog 条款，你应该明白：

Term	描述
Facility	The identifier used to describe the application or process that submitted the log message. Examples are mail, kernel, and ftp.
Priority	An indicator of the importance of the message. Levels are defined within syslog as guidelines, from debugging information to critical events.
Selector	A combination of one or more facilities and levels. When an incoming event matches a selector, an action is performed.
Action	What happens to an incoming message that matches a selector. Actions can write the message to a log file, echo the message to a console or other device, write the message to a logged in user, or send the message along to another syslog server.

系统日志设备：

这里是可用的设备选择。并非所有的设施都存在于所有版本的UNIX。

设备	描述
auth	Activity related to requesting name and password (getty, su, login)
authpriv	Same as auth but logged to a file that can only be read by selected users
console	Used to capture messages that would generally be directed to the system console
cron	Messages from the cron system scheduler
daemon	System daemon catch-all
ftp	Messages relating to the ftp daemon
kern	Kernel messages
local0.local7	Local facilities defined per site
lpr	Messages from the line printing system
mail	Messages relating to the mail system
mark	Pseudo event used to generate timestamps in log files
news	Messages relating to network news protocol (nntp)
ntp	Messages relating to network time protocol
user	Regular user processes
uucp	UUCP subsystem

Syslog优先级：

该系统记录的优先级总结在下面的表中：

优先级	描述
emerg	Emergency condition, such as an imminent system crash, usually broadcast to all users
alert	Condition that should be corrected immediately, such as a corrupted system database
crit	Critical condition, such as a hardware error
err	Ordinary error
warning	Warning
notice	Condition that is not an error, but possibly should be handled in a special way
info	Informational message
debug	Messages that are used when debugging programs
none	Pseudo level used to specify not to log messages.

设备和级别的组合，让你辨识记录和信息。

由于每个程序尽职地发送它的消息的系统记录器，记录器作出决定什么来跟踪和丢弃的基础上在选择器中定义的级别。

当你指定一个级别，系统将跟踪所有在这一水平较高。

/etc/syslog.conf 文件:

/etc/syslog.conf 文件控制，记录消息的位置。一个典型的 syslog.conf 文件可能看起来像这样：

```
*.err;kern.debug;auth.notice /dev/console
daemon,auth.notice           /var/log/messages
lpr.info                     /var/log/lpr.log
mail.*                       /var/log/mail.log
ftp.*                        /var/log/ftp.log
auth.*                       @prep.ai.mit.edu
auth.*                       root,amrood
netinfo.err                  /var/log/netinfo.log
install.*                    /var/log/install.log
*.emerg                      *
*.alert                      |program_name
mark.*                       /dev/console
```

文件的每一行包含两个部分：

- 消息选择器，指定哪种要记录的消息。例如，所有的错误消息或内核的所有调试信息。
- 应该做些什么消息，说一个动作域。例如，把它放在一个文件或消息发送到用户的终端上。

以下是对上述配置的显着点：

- 消息选择器有两个部分组成：设备和优先级。例如，kern.debug的选择由内核（设施）产生的所有调试消息（优先级）。
- 消息选择kern.debug的选择所有优先级大于调试。
- 设施或优先的地方中的星号表示“所有”。例如，. 调试是指所有调试信息，而 *kern.* 指由内核生成的所有消息。
- 您还可以使用逗号指定多个设备。两个或多个选择可以组合在一起使用分号。

日志操作：

action字段指定的五个动作之一：

1. 日志消息发送到一个文件或设备。例如，/var/log/lpr.log 或 /dev/console.。
2. 发送一条信息给用户。您可以指定多个用户名（如根，amrood）用逗号将它们分隔开。
3. 发送一条信息给所有用户。在这种情况下，“动作”字段中包含一个星号（例如，*）。
4. 管道消息的程序。在这种情况下，程序被指定后，UNIX管道符号（|）。
5. 将消息发送到另一台主机上的系统日志。在这种情况下，行动领域包括主机名，前面有一个at符号（例如，@ yiibai.com）

logger命令：

UNIX 提供了命令logger， 这是一个非常有用的命令处理系统日志。 logger 命令记录消息发送到syslogd守护进程，从而引发系统日志。

这意味着我们可以在命令行检查随时syslogd 守护进程，它的配置。 logger 命令提供系统日志文件，在命令行添加一行条目的方法。

该命令的格式是：

```
logger [-i] [-f file] [-p priority] [-t tag] [message]...
```

下面是详细的参数：

选项	描述
-f filename	Use the contents of file filename as the message to log.
-i	Log the process ID of the logger process with each line.
-p priority	Enter the message with the specified priority (specified selector entry); the message priority can be specified numerically, or as a facility.priority pair. The default priority is user.notice.
-t tag	Mark each line added to the log with the specified tag.
message	The string arguments whose contents are concatenated together in the specified order, separated by the space

您可以使用联机帮助帮助检查完成此命令的语法。

日志切换：

日志文件的增长倾向非常快，消耗大量的磁盘空间。要启用日志切换，大多数发行版使用 newsyslog 或 logrotate 工具。

这些工具应该被称为使用cron守护程序在频繁的时间间隔。检查newsyslog 或 logrotate的更多详细信息的手册页。

重要的日志位置

所有的系统应用程序创建日志文件在 /var/log 和其子目录。这里有几个重要的应用程序及其日志目录：

应用程序	目录
httpd	/var/log/httpd
samba	/var/log/samba
cron	/var/log/
mail	/var/log/
mysql	/var/log/

UNIX 信号和陷阱 - Unix

信号发送到一个程序来表示，一个重要的事件已经发生软件中断。事件可以从用户请求访问非法内存错误。一些信号，例如中断信号，表明用户提出的程序来完成的东西，而不是在通常的控制流。

以下是一些你可能会遇到的，要在程序中使用的更常见的信号：

信号名称	信号数	描述
SIGHUP	1	Hang up detected on controlling terminal or death of controlling process
SIGINT	2	Issued if the user sends an interrupt signal (Ctrl + C).
SIGQUIT	3	Issued if the user sends a quit signal (Ctrl + D).
SIGFPE	8	Issued if an illegal mathematical operation is attempted
SIGKILL	9	If a process gets this signal it must quit immediately and will not perform any clean-up operations
SIGALRM	14	Alarm Clock signal (used for timers)
SIGTERM	15	Software termination signal (sent by kill by default).

信号一览表：

有一个简单的方法，列出了所有你的系统支持的信号。只要发出 kill -l 命令，它会显示所有支持的信号：

```
$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGABRT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     16) SIGSTKFLT
17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU
25) SIGXFSZ     26) SIGVTALRM   27) SIGPROF     28) SIGWINCH
29) SIGIO       30) SIGPWR      31) SIGSYS      34) SIGRTMIN
35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3  38) SIGRTMIN+4
39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12
47) SIGRTMIN+13 48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14
51) SIGRTMAX-13 52) SIGRTMAX-12 53) SIGRTMAX-11 54) SIGRTMAX-10
55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7  58) SIGRTMAX-6
59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
```

实际的信号列表之间的Solaris，HP-UX 和 Linux。

默认操作：

每个信号都有默认与它相关联的动作。信号的默认操作是，当它接收信号的动作脚本或程序执行。

有些可能违反的操作是：

- 终止进程。
- 忽略信号。
- 核心转储。这将创建一个文件称为核心记忆形象的过程，当它收到的信号。
- 停止进程。
- 继续停止的进程。

发送信号：

有几种方法提供一个程序或脚本的信号。其中最常见的是执行脚本时用户键入Control-C或INTERRUPT 键。

当你按下Ctrl+ C键发送一个SIGINT的脚本，并按照定义的默认动作脚本终止。

其他常见的传送信号的方法是使用kill命令的语法如下：

```
$ kill -signal pid
```

signal 信号提供的电话号码或姓名，pid是信号要发送到的进程ID。示例：

```
$ kill -1 1001
```

HUB或挂起信号发送到正在运行的程序的进程ID 1001。发送kill信号相同的过程使用以下命令：

```
$ kill -9 1001
```

这会杀死正在运行的进程的进程ID 1001。

捕获信号：

当你按下Ctrl + C键或Break键在终端一个shell程序的执行过程中，正常程序将立即终止，并返回命令提示符。这可能并不总是可取的。例如，你可能最终留下了一堆临时文件，将不会清理。

捕获这些信号是很容易的，trap命令的语法如下：

```
$ trap commands signals
```

这里的命令可以是任何有效的Unix命令，或什至一个用户定义的函数，信号可以是任意数量的信号，你想来捕获的列表。

在shell脚本中的陷阱有三种常见的用途：

1. 清理临时文件
2. 忽略信号

清理临时文件：

trap命令作为一个例子，下面展示了如何可以删除一些文件，然后退出，如果有人试图从终端中止程序：

```
$ trap "rm -f $WORKDIR/work1$$ $WORKDIR/dataout$$; exit" 2
```

执行shell程序，这个陷阱的角度，这两个文件work1$$ 和 dataout$$ 将被自动删除，如果程序接收信号数为2。

因此，用户中断执行，如果执行的程序后，这个陷阱你可以放心，这两个文件将被清理。 exit 命令如下 rm 是必要的，因为没有它的执行将继续在节目中的一点，它离开时收到信号。

1号信号产生挂断：要么有人故意挂断线路或线路被意外断开。

您可以修改前面的陷阱也删除指定的文件，在这种情况下，两个信号信号1号添加到列表：

```
$ trap "rm $WORKDIR/work1$$ $WORKDIR/dataout$$; exit" 1 2
```

现在，这些文件将被删除，如果该行被挂了，或者按Ctrl+ C键被按下。

来捕获指定的命令必须用引号括起来，如果它们包含一个以上的命令。另外请注意，在 shell 命令行扫描 trap 命令得到执行，并再次当一个所列出的的信号被接收的时间。

WORKDIR 值 $$ 所以在前面的例子中，将被取代 trap 命令执行的时间。如果你想这种替代发生在收到信号1或2的时间你可以把单引号内的命令：

```
$ trap 'rm $WORKDIR/work1$$ $WORKDIR/dataout$$; exit' 1 2
```

忽略信号：

如果陷阱列出的命令是空的，指定的信号接收时，将被忽略。例如，下面的命令：

```
$ trap '' 2
```

指定的中断信号是被忽略的。你可能要忽略某些信号时进行一些操作，不希望打断。可以指定多个信号被忽略如下：

```
$ trap '' 1 2 3 15
```

注意，第一个参数必须被指定为一个信号被忽略，而不是相当于写入下面的内容，它具有独立的含义也各有：

```
$ trap  2
```

如果你忽略了一个信号，所有的子shell也忽略该信号。不过，如果指定要采取的行动在收到的信号，所有的子shell仍然会在收到该信号的默认操作。

重设陷阱：

当你改变了默认在收到信号后应采取的动作，你可以改变它回来的陷阱，如果你只是省略第一个参数;

```
$ trap 1 2
```

复位应采取的动作收到信号1或2 返回默认。

Unix 有用命令 - Unix

这里列出的命令，包括语法和简要说明。如需详细资料，请使用：

```
$man command
```

文件和目录：

这些命令允许你创建的目录和处理文件。

命令	描述
cat	Display File Contents
cd	Changes Directory to dirname
chgrp	change file group
chmod	Changing Permissions
cp	Copy source file into destination
file	Determine file type
find	Find files
grep	Search files for regular expressions.
head	Display first few lines of a file
ln	Create softlink on oldname
ls	Display information about file type.
mkdir	Create a new directory dirname
more	Display data in paginated form.
mv	Move (Rename) a oldname to newname.
pwd	Print current working directory.
rm	Remove (Delete) filename
rmdir	Delete an existing directory provided it is empty.
tail	Prints last few lines in a file.
touch	Update access and modification time of a file.

操作数据：

可以比较改变文件的内容，并用下面的命令。

命令	描述
awk	Pattern scanning and processing language
cmp	Compare the contents of two files
comm	Compare sorted data
cut	Cut out selected fields of each line of a file
diff	Differential file comparator
expand	Expand tabs to spaces
join	Join files on some common field
perl	Data manipulation language
sed	Stream text editor
sort	Sort file data
split	Split file into smaller files
tr	Translate characters
uniq	Report repeated lines in a file
wc	Count words, lines, and characters
vi	Opens vi text editor
vim	Opens vim text editor
fmt	Simple text formatter
spell	Check text for spelling error
ispell	Check text for spelling error
ispell	Check text for spelling error
emacs	GNU project Emacs
ex, edit	Line editor
emacs	GNU project Emacs
emacs	GNU project Emacs

压缩文件：

文件可能被压缩以节省空间。压缩文件，可以创建和检查：

命令	描述
compress	Compress files
gunzip	Uncompress gzipped files
gzip	GNU alternative compression method
uncompress	Uncompress files
unzip	List, test and extract compressed files in a ZIP archive
zcat	Cat a compressed file
zcmp	Compare compressed files
zdiff	Compare compressed files
zmore	File perusal filter for crt viewing of compressed text

获取信息：

各种UNIX手册和文档上线。下面的 Shell 命令给出的信息：

命令	描述
apropos	Locate commands by keyword lookup
info	Displays command information pages online
man	Displays manual pages online
whatis	Search the whatis database for complete words.
yelp	GNOME help viewer

网络通信：

这些命令是用来发送和接收文件从全球各地远程主机到本地UNIX主机。

命令	描述
ftp	File transfer program
rcp	Remote file copy
rlogin	Remote login to a UNIX host
rsh	Remote shell
tftp	Trivial file transfer program
telnet	Make terminal connection to another host
ssh	Secure shell terminal or command connection
scp	Secure shell remote file copy
sftp	secure shell file transfer program

出于安全原因，有些命令在您的计算机可能会受到限制。

用户之间的消息：

UNIX 系统支持的屏幕上的消息给其他用户和世界各地的电子邮箱：

命令	描述
evolution	GUI mail handling tool on Linux
mail	Simple send or read mail program
mesg	Permit or deny messages
parcel	Send files to another user
pine	Vdu-based mail utility
talk	Talk to another user
write	Write message to another user

编程工具：

下面的编程语言和工具都是基于你已经安装到你的Unix。

命令	描述
dbx	Sun debugger
gdb	GNU debugger
make	Maintain program groups and compile programs.
nm	Print program's name list
size	Print program's sizes
strip	Remove symbol table and relocation bits
cb	C program beautifier
cc	ANSI C compiler for Suns SPARC systems
ctrace	C program debugger
gcc	GNU ANSI C Compiler
indent	Indent and format C program source
bc	Interactive arithmetic language processor
gcl	GNU Common Lisp
perl	General purpose language
php	Web page embedded language
py	Python language interpreter
asp	Web page embedded language
CC	C++ compiler for Suns SPARC systems
g++	GNU C++ Compiler
javac	JAVA compiler
appletvieweir	JAVA applet viewer
netbeans	Java integrated development environment on Linux
sqlplus	Run the Oracle SQL interpreter
sqlldr	Run the Oracle SQL data loader
mysql	Run the mysql SQL interpreter

其他命令：

这些命令列出或改变有关系统的信息：

命令	描述
chfn	Change your finger information

chgrp	Change the group ownership of a file
chown	Change owner
date	Print the date
determin	Automatically find terminal type
du	Print amount of disk usage
echo	Echo arguments to the standard options
exit	Quit the system
finger	Print information about logged-in users
groupadd	Create a user group
groups	Show group memberships
homequota	Show quota and file usage
iostat	Report I/O statistics
kill	Send a signal to a process
last	Show last logins of users
logout	log off UNIX
lun	List user names or login ID
netstat	Show network status
passwd	Change user password
passwd	Change your login password
printenv	Display value of a shell variable
ps	Display the status of current processes
ps	Print process status statistics
quota -v	Display disk usage and limits
reset	Reset terminal mode
script	Keep script of terminal session
script	Save the output of a command or process
setenv	Set environment variables
stty	Set terminal options
time	Time a command
top	Display all system processes
tset	Set terminal mode

tty	Print current terminal name
umask	Show the permissions that are given to view files by default
uname	Display name of the current system
uptime	Get the system up time
useradd	Create a user account
users	Print names of logged in users
vmstat	Report virtual memory statistics
w	Show what logged in users are doing
who	List logged in users

Shell 内置数学函数 - Unix

本教程的一部分覆盖Bourne Shell的，但此页面列表下的所有内置的数学Korn Shell 中可用函数。

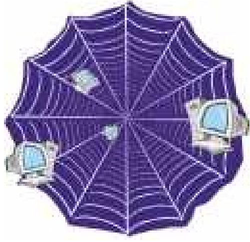
Korn Shell的提供一套标准的数学函数。他们被称为使用C 函数调用语法。

函数	描述
abs	Absolute value
log	Natural logarithm
acos	Arc cosine
sin	Sine
asin	Arc sine
sinh	Hyperbolic sine
cos	Cosine
sqrt	Square root
cosh	Hyperbolic cosine
tan	Tangent
exp	Exponential function
tanh	Hyperbolic tangent
int	Integer part of floating-yiibai number

网站建设教程

网站建设指南

WWW 指南-万维网联盟(World Wide Web)



WWW - 万维网联盟

WWW通常称为网络。web是一个世界各地的计算机网络。电脑在Web上使用标准语言沟通。万维网联盟（W3C）制定了Web标准

什么是WWW？

- WWW 代表 **World Wide Web(万维网)**
- 万维网常常被称为 网络
- 网络是世界各地的计算机网络
- 网络中的所有电脑可以相互沟通
- 所有的计算机使用HTTP的通信标准

万维网如何工作？

- 信息存储的文件称为网页
- Web页面存储在Web服务器上。
- 阅读网页的计算机被称为Web客户端
- Web客户端查看网页的程序称为Web浏览器
- 行的浏览器有Internet Explorer，Chrome和火狐等。

浏览器如何获取网页？

- 一个浏览器通过请求从服务器上获取的网页数据
- 标准的HTTP请求包含一个网页地址
- 网页地址实例: **http://www.w3cschool.cc**

浏览器如何显示页面？

- 所有的网页包含如何显示的说明
- 浏览器通过过阅读这些说明显示页面。
- 最常见的显示指令被称为HTML标签
- HTML中段落的标签为： **<p>**
- 在HTML中，段落是这样定义的: **<p>**这是段落。**</p>**

是谁在做**Web**标准？

- Web标准并不是谷歌或微软在做
- 制定Web规则的机构是W3C
- W3C表示万维网联盟
- W3C 制定的web标准规范。
- 最重要的Web标准是HTML，CSS和XML

HTML 指南

HTML - 超文本标记语言 (Hyper Text Markup Language)

HTML是建设网站/网页制作主要语言。

HTML是一种易于学习的标记语言。

HTML使用像 <p> 尖括号内标记标签来定义网页的内容：

HTML 实例

```
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

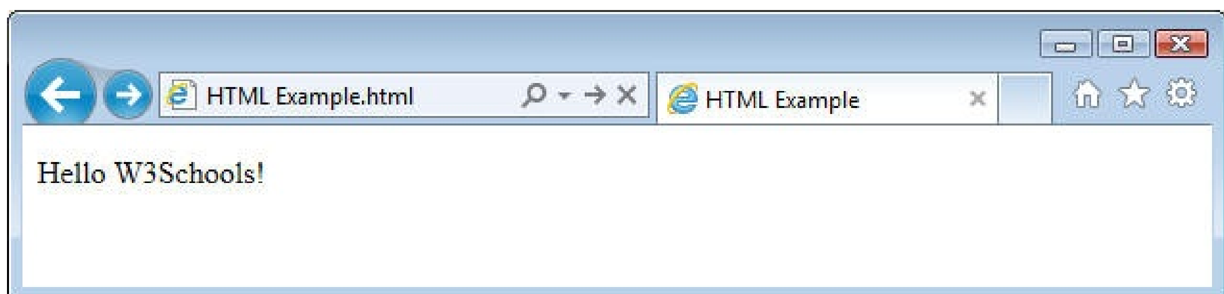
</body>
</html>
```

尝试一下？点击 "尝试一下" 按钮查看实例。

HTML使用开始标记和结束标记来标记一个网页元素， ：在上面的例子，<p>标签标志着一个段落开始，</p>标志着该段末尾。

通过使用简单的HTML标签，网页设计师可以为一个网页（HTML文档）添加标题，段落，文字，表格，图片，列表，编程代码等。

Web浏览器（IE浏览器，火狐，Chrome等）读取HTML文档，解释HTML标记，并显示正确用户可读的内容（不显示HTML标签）：



根据HTML标准，HTML可用于定义网页的内容。/p>

要定义视觉样式（颜色，大小，外观，布局等），应使用CSS（层叠样式表）（见下一章）。

如何 学习 HTML?

W3CSchool提供了完整的HTML教程，你可以学到所有关于HTML的知识。

HTML学习简单 - 你会喜欢上它的。

学习我们的 完整的 HTML 教程

学习我们的 完整的 HTML 参考手册

CSS 指南

CSS - 层叠样式表(Cascading Style Sheets)

CSS定义如何显示HTML元素。

CSS 描述了HTML元素的可视化样式(外观，布局，颜色，字体)。

CSS是单独设计的文件（从而大大提高HTML的灵活性和减少HTML的复杂性）。

CSS简单易学。你可以把一个HTML元素当作选择器，并在大括号内的列出样式属性

CSS 实例

```
body
{
background-color:#d0e4fe;
}
h1
{
color:orange;
text-align:center;
}
p
{
font-family:"Times New Roman";
font-size:20px;
}
```

尝试一下 ?

点击 "尝试一下" 按钮，查看在线实例

样式表极大地提高了工作效率

样式表定义如何显示 HTML 元素，诸如 HTML 3.2 的样式中的字体标签和颜色属性通常被保存在外部的 .css 文件中。

通过仅仅编辑一个简单的 CSS 文档，外部样式表使你有能力同时改变站点中所有页面布局的外观。

由于允许同时控制多重页面的样式和布局，CSS 可以称得上 WEB 设计领域的一个突破。作为网站开发者，你可以为每个 HTML 元素定义样式，并将之应用于你希望的任意多的页面中。如需进行全局变换，只需简单地改变样式，然后网站中的所有元素均会被自动地更新。

另一个 CSS 实例

如何学习 CSS?

学习我们的 完整的 CSS 教程

学习我们的 完整的 CSS 参考手册

JavaScript 指南

JavaScript - 客户端脚本

JavaScript 是属于网络的脚本语言!

JavaScript 被数百万计的网页用来改进设计、验证表单、检测浏览器、创建cookies,以及更多的应用。

JavaScript 学习简单

JavaScript 实例

```
<h2>My First Web Page</h2>
<p id="demo">This is a paragraph.</p>
<button type="button">Display Date</button>
```

尝试一下？点击 "尝试一下" 按钮查看在线实例。

什么是 JavaScript?

- JavaScript 被设计用来向 HTML 页面添加交互行为。
- JavaScript 是一种脚本语言（脚本语言是一种轻量级的编程语言）。
- JavaScript 由数行可执行计算机代码组成。
- JavaScript 通常被直接嵌入 HTML 页面。
- JavaScript 是一种解释性语言（就是说，代码执行不进行预编译）。
- 所有的人无需购买许可证均可使用 JavaScript。

客户端脚本

JavaScript "制定" 浏览器行为。这就是所谓的客户端脚本（或浏览器的脚本）。

服务器端脚本是"制定"服务器的行为（见本站的ASP / PHP教程）。

JavaScript可以做什么？

- **JavaScript 为 HTML 设计师**提供了一种编程工具 HTML 创作者往往都不是程序员，但是 JavaScript 却是一种只拥有极其简单的语法的脚本语言！几乎每个人都有能力将短小的

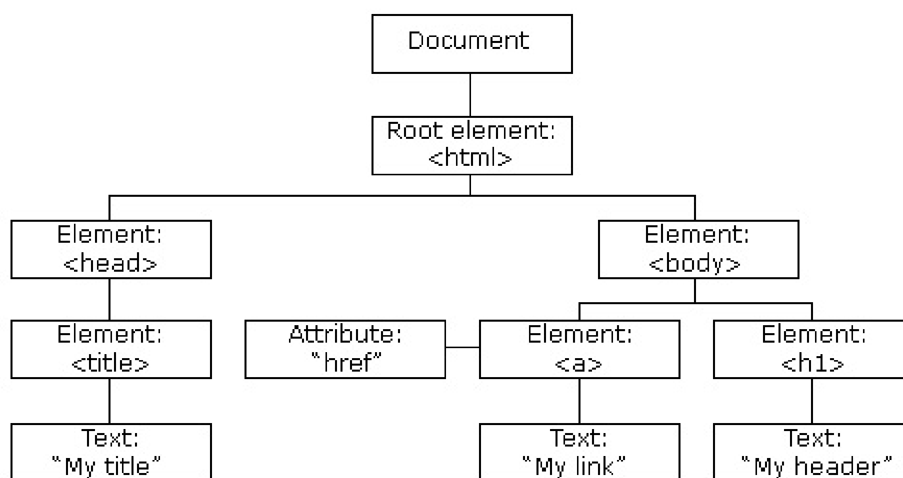
代码片断放入他们的 HTML 页面当中。
- **JavaScript** 可以将动态的文本放入 **HTML** 页面 类似于这样的一段 JavaScript 声明可以将一段可变的文本放入 HTML 页面：document.write("<h1>" + name + "</h1>")
- **JavaScript** 可以对事件作出响应 可以将 JavaScript 设置为当某事件发生时才会被执行，例如页面载入完成或者当用户点击某个 HTML 元素时。
- **JavaScript** 可以读写 **HTML** 元素 JavaScript 可以读取及改变 HTML 元素的内容。
- **JavaScript** 可被用来验证数据 在数据被提交到服务器之前，JavaScript 可被用来验证这些数据。
- **JavaScript** 可被用来检测访问者的浏览器 JavaScript 可被用来检测访问者的浏览器，并根据所检测到的浏览器，为这个浏览器载入相应的页面。
- **JavaScript** 可被用来创建 **cookies** JavaScript 可被用来存储和取回位于访问者的计算机中的信息。

什么是HTML DOM？

HTML DOM 定义了访问和操作 HTML 文档的标准方法。

DOM 将 HTML 文档表达为树结构。

HTML DOM Tree 实例



如何学习JavaScript？

访问完整的 JavaScript 教程

访问 完整的 HTML DOM 教程

访问完整的 JavaScript 和 HTML DOM 参考手册

XML 指南

XML - 可扩展标记语言(EXtensible Markup Language)

XML 是跨平台的、用于传输信息且独立于软件和硬件的工具。

XML 文档实例

```
<?xml version="1.0"?><note>    <to>Tove</to>    <from>Jani</from>    <heading>Reminder</h
```

什么是XML？

- XML 指可扩展标记语言（EXtensible Markup Language）
- XML 是一种标记语言，很类似 HTML
- XML 被设计用来描述数据
- XML 标签没有被预定义。您需要自行定义标签。
- XML 使用文件类型声明（DTD）或者 *XML Schema* 来描述数据。
- 带有 DTD 或者 XML Schema 的 XML 被设计为具有自我描述性。
- XML 是一个 W3C 标准

XML不会做任何事情

ML是不做任何事情。 XML创建结构，存储和携带信息。

上面的XML文档的例子是XML编写的从Jani到Tove的一张纸条。注意标题和邮件正文。它还具有来自哪里的信息。但是，这个XML文档并没有做任何事情。只是纯粹的信息包裹在XML标记中。必须有人写了一款软件发送，接收或显示它：

MESSAGE**To: **Tove** From:** JaniDon't forget me this weekend!

XML标签不是预定义

XML标签不是预定义，您必须"发明"自己的标签。

用来标记HTML文档的标签是预定义的的HTML文件作者只能使用在HTML标准（如<P>，<H1>等）定义的标签。

XML允许作者来定义他/她自己的标签和他/她自己的文档结构。

在上面的例子（像<to>和<from>）标签没有在任何XML标准定义。这些标签是XML文档作者"发明"的。

查看一个XMLCD目录

查看一个XML植物目录

查看一个XML食品菜单

如何学习XML？

学习我们完整的XML教程

服务端脚本 指南

ASP 和 PHP - 服务端脚本

HTML 文件可以包含文本、HTML 标签以及脚本。

服务器端脚本是对服务器行为的编程。这被称为服务器端脚本或服务器脚本。

客户端脚本是对浏览器行为的编程。（请参阅 JavaScript 初级教程）。

通常，当浏览器请求某个 HTML 文件时，服务器会返回此文件，但是假如此文件含有服务器端的脚本，那么在此 HTML 文件作为纯 HTML 被返回浏览器之前，首先会执行 HTML 文件中的脚本。

服务器脚本能做什么呢？

- 动态地向 web 页面编辑、改变或添加任何的内容
- 对由 HTML 表单提交的用户请求或数据进行响应
- 访问数据或数据库，并向浏览器返回结果
- 为不同的用户定制页面
- 提高网页安全性，使您的网页代码不会通过浏览器被查看到

重要提醒： 由于脚本在服务器上执行，因此浏览器在不支持脚本的情况下就可以显示服务器端的文件！

ASP 和 PHP

在 W3CSchool，我们通过使用活动服务器页面（ASP）和超文本预处理器（PHP）来演示服务器端的脚本编程。

ASP 实例

通过 ASP 来写文本 何通过 ASP 来写文本。

```
<!DOCTYPE html>
<html>
<body>

<%
response.write("Hello World!")
%>

</body>
</html>
```

向文本添加 HTML 如何通过 HTML 标签来格式化文本。

```
<!DOCTYPE html>
<html>
<body>
<%
response.write("<h2>You can use HTML tags to format the text!</h2>")
%>

<%
response.write("<p style='color:#0000ff'>This text is styled with the style attribute!</p
%>
</body>
</html>
```

如何学习ASP或PHP？

学习我们完整的 ASP 教程, 或者我们 的 完整的 PHP 教程。

SQL 指南

SQL - 结构化查询语言 (Structured Query Language)

SQL 是用于访问和处理数据库的标准的计算机语言。

常用的数据库管理系统： MySQL, SQL Server, Access, Oracle, Sybase, 和 DB2

对于那些希望在数据库中存储数据并从中获取数据的人来说，SQL 的知识是价值无法衡量的。

什么是 SQL?

- SQL 指结构化查询语言 (_S_tructured _Q_uery _L_anguage)
- SQL 使我们有能力访问数据库
- SQL 是一种 ANSI 的标准计算机语言
- SQL 面向数据库执行查询
- SQL 可从数据库取回数据
- SQL 可在数据库中插入新的记录
- SQL 可从数据库删除记录
- SQL 很容易学习

SQL 数据库表

一个数据库通常包含一个或多个表。每个表由一个名字标识（例如"客户"或者"订单"）。表包含带有数据的记录（行）。

下面的例子是一个名为 "Persons" 的表：

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger

上面的表包含三条记录（每一条对应一个人）和四个列（姓、名、地址和城市）。

SQL 查询程序

通过 SQL，我们可以查询某个数据库，并获得返回的一个结果集。

查询程序类似这样：

SELECT LastName FROM Persons

结果集类似这样：

LastName
Hansen
Svendson
Pettersen

如何学习SQL？

访问我们 完整的 SQL 教程

Web 创建设计

设计一个网站，需要认真思考和规划。

最重要的是要知道你的访问用户。

用户是浏览者

一个典型的访问者将无法读取您的网页的全部内容！

无论您在网页中发布了多么有用的信息，一个访问者在决定是否继续阅读之前仅仅会花几秒钟的时间进行浏览。

请确保使你的观点，在页面的第一句！另外，您还需要在整个页面中使用简短的段落以及有趣的标题。

少即是多

保持段落尽可能短。

保持章节尽可能短。

冗长文字的页面不利于用户体验。

如果你的网页内容很多，您将页面信息分解成小的模块，并放置在不同的页面！

导航

在您网站的所有页面使用一致的导航结构。

不要在文本段落内使用超链，超链接会把访问者带到别的页面，这样做会破坏导航结构一致性。

如果您必须使用超链接，你可以将链接添加到一个段落的底部或菜单中。

加载速度

有时开发人员不知道一些网页需要很长的时间来加载。

据统计，大多数用户会留在加载时间不超过7秒的网页。

测试您的网页在一个低速的调制解调器中打开。如果您的网页需要很长时间加载，可以考虑删除图片或多媒体等内容。

用户反馈

反馈是一件非常好的事情！

你的访问者是你的"客户"。通常他们会给你的网站提供很好的改善建议。

如果您提供良好的反馈途径，您将得到来自很多来自不同领域人的反馈意见。

访问者的显示器

在互联网上不是每个人的显示器尺寸是不一样的。

如果你设计一个网站，是用高分辨率的显示器上显示，当分辨率低的显示器（如800 × 600）访问你的网页时就可能会出现问题。

请在不同的显示器上测试您的网站。

查看我们的 显示器 了解显示器的发展趋势。

他们使用什么浏览器？

请在不同的浏览器测试你的网站。

目前最流行的浏览器有：Internet Explorer，Firefox和Google Chrome。

设计网页时，一个明智的做法是使用正确的HTML。正确的编码将帮助浏览器正确显示您的网页。

访问我们的 浏览器统计 信息了解浏览器的发展趋势。

客户端使用的插件

声音，视频剪辑，或其他多媒体内容可能需要使用单独的程序（插件）来播放。

请确保您的访问者能在你的网页上正常使用他们所需要的软件。

关于残疾人呢？

有些人的视力或听力有障碍。

们可能会尝试使用盲文或语音浏览器浏览您的网页。所以你应该在你的网页添加图像和图形元素的替代文本。

Web 标准

Web标准，使得Web开发更加容易。

Web标准由万维网联盟（W3C）制定。

为什么要Web标准？

对于浏览器开发商和web程序开发人员在开发新的应用程序时遵守指定的标准更有利于web更好地发展。

开发人员按照Web标准制作网页，这样对于开发者来说就更加简单了，因为他们可以很容易了解彼的编码

使用Web标准，将确保所有浏览器正确显示您的网站而无需费时重写。

遵守标准的Web页面可以使得搜索引擎更容易访问并收入网页，也可以更容易转换为其他格式，并更易于访问程序代码（如JavaScript和DOM）。

提示： 你可以使用网页验证服务器验证页面的标准性。

无障碍

无障碍环境是一个HTML标准的重要组成部分。

Web标准，使其更易于为残疾人士使用Web。

Web标准使得残疾人士也可以很容易地使用互联网。盲人可使用程序为他们读出网页。而弱视的人群可通过重新排列并放大网页来访问网站。

W3C - 万维网联盟

W3C 创建和维护Web标准。

姆·伯纳斯·李（Tim Berners-Lee）是万维网联盟创始人发明者被称为互联网之父：

"The dream behind the Web is of a common information space in which we communicate by sharing information."

万维网联盟，建立于 1994 年，是一个国际性的联盟，其宗旨是投身于"引领 web 以激发其全部潜能"。

- W3C表示万维网联盟
- W3C创建于**1994**年**10**月
- W3C被**Web**发明者蒂姆·伯纳斯·李（**Tim Berners-Lee**）创建
- W3C是作为成员国机构组织
- W3C的工作是进行标准化网络
- W3C创建和维护的**WWW**标准
- W3C标准有**W3C**建议

最重要W3C标准有：

- HTML
- CSS
- XML
- XSL
- DOM

W3C官方主页

ECMA - 欧洲计算机制造商协会（European Computer Manufacturers Association）

ECMA于1960年在布鲁塞尔由一些欧洲最大的计算机和技术公司成立。到1961年5月，他们成立了一个正式的组织，这个组织的目标是评估，开发和认可电信和计算机标准。

大家决定把ECMA的总部设在日内瓦是因为这样能够让它与其它与之协同工作的标准制定组织更接近一些，比方说国际标准化组织（ISO）和国际电子技术协会（IEC）。

ECMA是"European Computer Manufactures Association"的缩写，中文称欧洲计算机制造联合会。是1961年成立的旨在建立统一的电脑操作格式标准--包括程序语言和输入输出的组织。

最新ECMAScript规范就是ECMA- 262：

http://www.ecma-international.org/publications/standards/Ecma-262.htm

Web 语义化

单词语义化表示了它的意义。

事物的语义化意味着事物。

Web 语义化 = Web的意义。

什么是 Web 语义化？

什么是语义化？其实简单说来就是让机器可以读懂内容。

- 甲壳虫乐队是一个来自利物浦受欢迎的乐队。
- 约翰列侬是披头士乐队的成员。
- "Hey Jude"是由披头士的代表作。

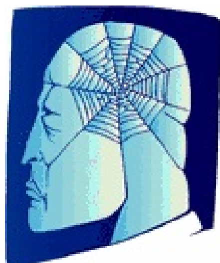
我们可以很容易理解上面的句子的意义。但这些语句怎么 被计算机理解呢？

语句由语法规则创建。语言的语法定义了创建语言语句的规则。但是如何让语法变为语义呢？

语义网是让机器可以理解数据。语义网技术，它包括一套描述语言和推理逻辑。它包通过一些格式对本体（Ontology）进行描述。

语义网并不是网页之间的链接。

语义网描述了事物之间的关联(（如 A 是 B的一部分，Y 是 Z 的成员)及事物的属性（如大小，高度，年龄，价格等）。



语义网的实现是基于XML（可扩展标记语言eXtensible Markup Langauge）语言和资源描述框架（RDF）来完成的。XML是一种用于定义标记语言的工具，其内容包括XML声明、用以定义语言语法的DTD (document type declaration文档类型定义）、描述标记的详细说明以及文档本身。而文档本身又包含有标记和内容。RDF则用以表达网页的内容。

资源描述框架

RDF（Resource Description Framework），即资源描述框架，是W3C推荐的用来描述WWW上的信息资源及其之间关系的语言规范。

RDF（S）是语义网的重要组成部分，它使用URI来标识不同的对象（包括资源节点、属性类或属性值）并可将不同的URI连接起来，清楚表达对象间的关系。

实现

语义网虽然是一种更加美好的网络，但实现起来却是一项复杂而浩大的工程。 目前语义网的体系结构正在建设中，主要需要以下两方面的支持：

（1） 数据网络的实现

即：通过一套统一的完善的数据标准对网络信息进行更彻底更详细的标记，使得语义网能够精准的识别信息，区分信息的作用和含义 要使语义网搜索更精确彻底，更容易判断信息的真假，从而达到实用的目标，首先需要制订标准，该标准允许用户给网络内容添加元数据（即解释详尽的标记），并能让用户精确地指出他们正在寻找什么；然后，还需要找到一种方法，以确保不同的程序都能分享不同网站的内容；最后，要求用户可以增加其他功能，如添加应用软件等。

语义网的实现是基于XML（可扩展标记语言eXtensible Markup Langauge）语言和资源描述框架（RDF）来完成的。XML是一种用于定义标记语言的工具，其内容包括XML声明、用以定义语言语法的DTD (document type declaration文档类型定义）、描述标记的详细说明以及文档本身。而文档本身又包含有标记和内容。RDF则用以表达网页的内容。

（2）具有语义分析能力的搜索引擎

如果说数据网络能够短时间通过亿万的个体实现，那么网络的语义化智能化就要通过人类尖端智慧群体的努力实现。研发一种具有语义分析能力的信息搜索引擎将成为语义网的最重要一步，这种引擎能够理解人类的自然语言，并且具有一定的推理和判断能力。

语义搜索引擎（semantic search engine）和具有语义分析能力的搜索引擎(semantically enabled search engine)是两码事。前者不过是语义网络的利用，一种信息搜索方式，而具有语义分析能力的搜索引擎是一种能够理解自然语言，通过计算机的推理而进一步提供更符合用户心理的答案。

前景

语义网的体系结构正在建设中，当前国际范围内对此体系结构的研究还没有形成一个令人满意的严密的逻辑描述与理论体系，中国学者对该体系结构也只是在国外研究的基础上做简要的介绍，还没有形成系统的阐述。

语义网的实现需要三大关键技术的支持：XML、RDF和Ontology。

XML(eXtensible Marked Language，即可扩展标记语言）可以让信息提供者根据需要，自行定义标记及属性名，从而使XML文件的结构可以复杂到任意程度。

它具有良好的数据存储格式和可扩展性、高度结构化以及便于网络传输等优点，再加上其特有的NS机制及XML Schema所支持的多种数据类型与校验机制，使其成为语义网的关键技术之一。

目前关于语义网关键技术的讨论主要集中在RDF和Ontology身上。

RDF是W3C组织推荐使用的用来描述资源及其之间关系的语言规范，具有简单、易扩展、开放性、易交换和易综合等特点。

值得注意的是，RDF 只定义了资源的描述方式，却没有定义用哪些数据描述资源。RDF由三个部分组成：RDF Data Model、RDF Schema和RDF Syntax。

附上：

1.语义网通过扩展现有的互联网，在信息中加入表示其含义的内容，使计算机可以自动与人协同工作。也就是说，语义网中的各种资源不再只是各种相连的信息，还包括其信息的真正含义，从而提高计算机处理信息的自动化和智能化。当然，计算机并不具有真正的智能，语义网的建立需要研究者们对信息进行有效的表示，制定统一的标准，使计算机可以对信息进行有效的自动处理。

（来源：何斌 张立厚《信息管理原理与方法》 清华大学出版社 2007年7月第二版）



语义网体系结构

- 第一层：Unicode与URI，是整个体系结构的基础。
- 第二层：XML+NS+XMLSchema,负责语法上表示数据的内容和结构，通过使用标准的格式语言将网络信息的表现形式、数据结构和内容分离。
- 第三层：RDF+RDF Schema,它提供语义模型用于描述网上的信息和类型。其中，RDF（Resource Description Framework），即资源描述框架，是W3C推荐的用来描述WWW上的信息资源及其之间关系的语言规范。RDF（S）是语义网的重要组成部分，它使用URI来标识不同的对象（包括资源节点、属性类或属性值）并可将不同的URI连接起来，清楚表达对象间的关系。
- 第四层：本体词汇层，本体是关于领域知识的概念化、形式化的明确规范。在语义网体系结构中，本体的作用主要表现在：（1）.概念描述，即通过概念描述揭示领域知识；（2）.语义揭示，本体具有比RDF更强的表达能力，可以揭示更为丰富的语义关系；（3）.一致性，本体作为领域知识的明确规范，可以保证语义的一致性，从而彻底解决一词多义、多词一义和词义含糊现象；（4）. 推理支持，本体在概念描述上的确定性及其强大的语义揭示能力在数据层面有力地保证了推理的有效性。
- 第五层：逻辑层，负责提供公理和推理原则，为智能服务提供基础。其中，描述逻辑(DescriptionLogic)是基于对象的知识表示的形式化，它吸取了KL-ONE的主要思想，是一阶谓词逻辑的一个可判定子集。它与一阶谓词逻辑不同的是，描述逻辑系统能提供可判定的推理服务。除了知识表示以外，描述逻辑还用在其它许多领域，它被认为是以对象为中心的表示语言的最为重要的归一形式。描述逻辑的重要特征是很强的表达能力和可判定性，它能保证推理算法总能停止，并返回正确的结果。在众多知识表示的形式化方法中，描述逻辑在十多年来受到人们的特别关注，主要原因在于：它们有清晰的模型-理论机制；很适合于通过概念分类学来表示应用领域；并提供了很用的推理服务。
- 第六层证明层和第七层信任层负责提供认证和信任机制。

Web Glossary

以下词汇按字母顺序排列

Access (Microsoft Access)

一个微软开发的数据库系统。 Microsoft Office专业版一部分。主要用于在Windows平台上运行低流量的Web网站。

ActiveMovie

微软公司推出的用于多媒体程序设计的控件

ActiveX

允许Web浏览器下载并执行Windows程序编程接口（API）。 （另见插件）

Address

请参阅网址。

AdSense

由Google提供的网络广告系统。

AJAX (Asynchronous JavaScript and XML)

使用JavaScript和XML来创建交互式Web应用程序的"艺术"。使用Ajax，Web应用程序可以在后台Web服务器（异步）交换数据和更新一个网页部分无需重新加载页面。

了解更多有关AJAX的信息在我们的AJAX教程

Anchor

在网络方面：起点或结束的超链接。

了解更多关于链接的信息在我们的HTML教程

Adobe Air

一个Adobe集成运行时（AIR）系统，使开发人员可以使用Web技术（HTML，JAVASCRIPT，Fash）来创建桌面应用程序。

Android

一款手机操作系统Android公司开发，后来被谷歌收购。

匿名 **FTP**

请参阅FTP服务器。

ANSI (American National Standards Institute)

为计算机行业创建标准的组织。 ANSIC标准负责。

ANSI C

C编程语言国际标准。

ADO (ActiveX Data Object)

微软的任何类型的数据存储技术，提供数据访问。

了解更多有关ADO的信息在我们的ADO教程

ADSL (Asymmetric Digital Subscriber Line)

一个特殊类型的DSL线路，上传速度和下载速度是不同的。

Agent

请参阅搜索代理/搜索引擎

Amaya

来自W3C的一个开放源码的Web浏览器编辑器，用于推动在浏览器设计领先的想法。

Animation

一组照片播放时模拟运动串联。

反病毒程序

一种计算机程序，发现并摧毁所有类型的计算机病毒。

Apache

一个开源的Web服务器。大多用于Unix，Linux和Solaris平台。

Applet

请参阅web小程序。

Archie

计算机程序来定位公共FTP服务器上文件。

API (Application Programming Interface)

让程序与另一个程序通信接口。在web方面：让网页浏览器或Web服务器与其他程序通信的接口。 （亦见Active - X的插件）

ARPAnet

网络测试实验，在20世纪70年代互联网发展时开始。

Authentication

在web方面：用什么方法在网络上或计算机程序上来验证用户的身份。

ASCII (American Standard 兼容 Information Interchange)

128位字母数字和特殊控制字符用于计算机存储和打印的文字。 HTML中通过web传输的数据。

在我们HTML参考中，参阅ASCII码完整列表

ASF (Advanced Streaming Format)

一个多媒体数据流的格式。微软的Windows Media开发。

ASP (Active Server Pages)

微软的技术，使服务器可执行的脚本在网页中插入。

了解更多关于ASP的信息访问我们的ASP教程

ASX (ASF Streaming Redirector)

关于ASF文件存储信息的XML格式。微软的Windows Media开发。

AVI (Audio Video Interleave)

文件格式的视频文件。微软开发的视频压缩技术。

Banner Ad

（最常见的图形）广告放置在网页上作为广告客户的网站超链接行为。

Bandwidth

您可以通过发送Internet连接速度（数据量）测量。更多的带宽更快的连接

Baud

通道上每秒钟发送符号数。

BBS (Bulletin Board System)

基于网络的公共分享讨论，文件和公告系统。

Binary Data

于机器可读的形式数据。

Bit (Binary Digit)

于计算机中存储的数据的最小单位。一个位值为0或1。一台计算机使用8位来存储一个文本字符。

Blog (Web Log)

一个网站类型（通常是由个人维护）日志评论（大多数通常是个人）的意见，含义说明事件等

Blogger

一个人维持或写入内容到网络日志（博客）。

Blogging

编写或内容添加到网络日志（博客）。

BMP (Bitmap)

用于存储图像的格式。

Bookmark

在web方面：一个特定的网站链接存储（书签）以备将来使用Web用户方便地访问。

Bounce Rate

网站访问者查看只有一个网页在他们离开（弹起）的比例。

Browse

术语来描述整个网络用户的运动通过超链接从一页一页地移动使用网页浏览器。（请参阅Web浏览器）。

BPS (Bits Per Second)

用术语来描述在网上进行数据传输速度。

Browser

请参阅Web浏览器。

Byte (Binary Term)

计算机存储单元包含8位。每个字节可以存储一个文本字符。

C

一种先进的编程语言用于先进的计算机应用编程。

C++ (C Plus Plus)

同样的c补充面向对象的功能。

C# (C Sharp)

微软的版本C ++的补充类似Java的功能。

Case Sensitive

用来描述使用大写或小写字母敏感

Cache

在web方面：一个Web浏览器或Web服务器计算机的硬盘上存储的网页副本功能。

Chat

on-line基于文本的互联网用户之间通信。

CGI (Common Gateway Interface)

描述一个CGI程序与Web服务器如何通信的规则。

CGI Bin

在Web服务器上存储CGI程序文件夹（或目录）。

CGI Program

一个小程序，从Web服务器处理输入和输出。 CGI程序通常用于处理表单输入或数据库查询。

Cinepac

计算机视频编解码器

Client

请参阅Web客户端。

Client/Server

在web方面：和Web客户端和Web服务器之间工作量分离的通信。

Click

在web方面：鼠标点击一个超链接元素（如文字或图片）在网页上创建一个事件如要到另一个网页访问或访问同一页其他部分。

Clickthrough Rate

访问者点击页面上的超链接（或广告）页面已经显示时间数的比例。

Cloud Computing

在互联网上存储的应用程序和数据（而不是用户的计算机上）。

Codec (Compressor / Decompressor)

用于数据压缩和解压技术通用术语。

Communication Protocol

标准（语言和一套规则），让电脑用一个标准方式进行交互。例如IP，FTP和HTTP。

了解更多关于通信协议的信息去我们的TCP / IP教程

Compression

更快通过网络交付Web文档或图形大小（压缩）方法。

Computer Virus

一种计算机程序，可能损害消息显示，删除文件甚至摧毁计算机操作系统的计算机。

Cookie

Web浏览器由Web服务器，在您的计算机上存储的信息。 Cookie的目的就是提供有关您访问网站的信息，在随后的访问服务器使用。

ColdFusion

Web开发软件的大多数平台（LINUX，UNIX，Solaris和Windows）。

CSS (Cascading Style Sheets)

一个W3C推荐定义Web文档样式（如字体，大小，颜色，间距等）语言。

了解更多关于CSS的信息于我们的CSS教程

Database

以这样的方式，一个计算机程序可以轻松地检索和操纵数据计算机存储数据。

了解更多关于数据库的信息于我们的SQL教程

Database System

操纵数据库中的数据的计算机程序（如MS ACCESS，Oracle和MySQL）。

DB2

来自IBM数据库系统。大多用于Unix和Solaris平台。

DBA (Data Base Administrator)

人（或软件）谁管理数据库。典型的任务：备份，维护和执行。

DHCP (Dynamic Host Configuration Protocol)

互联网标准协议，为NEE用户分配新的IP地址。

DHTML (Dynamic HTML)

常用的一个术语来描述HTML内容可能动态改变。

Dial-up Connection

在web方面：一个通过电话和调制解调器连接到Internet。

Discussion Group

请参阅Newsgroup。

DNS (Domain Name Service)

计算机程序运行在Web服务器上域名翻译成IP地址。

了解更多有关DNS的信息于我们虚拟主机教程

DNS Server

Web服务器执行DNS。

DOM (Document Object Model)

一个网页对象编程模型。（请参阅HTML DOM和XML DOM）

Domain Name

一个网站的名称标识。（如：w3cschool.cc）

了解更多有关域的信息于我们虚拟主机教程

DOS (Disk Operating System)

一个通用基于磁盘的计算机操作系统（见操作系统）。最初是微软IBM个人电脑。通常用于为MS - DOS简写。

Download

从远程计算机传输文件到本地计算机。在web方面：Web客户端从Web服务器传输文件。（另请参阅上传）。

DSL (Digital Subscriber Line)

基于普通电话线的宽带接入技术

DTD (Document Type Definition)

一个定义类似HTML或XMLweb文件合法构建模块的规则（一种语言）。

了解更多有关DTD的信息于我们DTD教程

Dynamic IP

每次连接到互联网的IP地址的变化。 （请参阅DHCP和静态IP）。

E-mail (Electronic Mail)

由一个人到另一个通过互联网发送消息。

E-mail Address

发送电子邮件给一个人或一个组织使用的地址。典型格式就是用户名@主机名。

E-mail Server

一个Web服务器专用电子邮件服务任务。

Encryption

要从原来的形式转换的数据，一个只能读取人可以逆转的加密形式。加密的目的是为了防止未经授权的读取数据。

Error

请参阅Web服务器错误。

Ethernet

一个局域网络类型（见局域网）。

Firewall

作为一个安全过滤器，可以限制类型的网络通信行为软件。最常用于个人计算机（或LAN）和互联网之间。

Flash

基于矢量的多媒体格式在Web上的使用，由Adobe开发

Form

请参阅HTML表单。

Forum

在web方面：同样的作为Newsgroup。

Frame

在web方面：浏览器中显示屏幕一部分特定内容。帧通常用于显示不同的网页内容。

FrontPage

Windows平台Web开发软件。由微软开发。

FTP (File Transfer Protocol)

两台计算机之间发送文件最常用的方法之一。

FTP Server

一个Web服务器，您可以登录，并下载文件（或文件上传到）。匿名FTP是一种不使用登录帐户从FTP服务器下载文件方法。

Gateway

一个计算机程序之间不兼容的应用程序或网络传输（格式化）数据。

GIF (Graphics Interchange Format)

一个由CompuServe开发的存储图像压缩格式。互联网上最常见的图像格式之一。

GB

Gigabyte，千兆字节，电脑的一种存储单位。

Gigabyte

1024兆字节。通常向下舍入到10亿字节。

Graphics

在web方面介绍图片（相对文本）。

Graphic Monitor

一个显示屏，可以显示图形。

Graphic Printer

一台打印机，可以打印图形。

Graphical Banner

请参阅Banner Ad。

Helper application

在web方面：一个方案帮助浏览器显示，视图或工作，浏览器本身不能处理。（请参阅插件）。

Hits

Web对象（网页或图片）已被浏览或下载的数量。（请参阅页点击）。

Home Page

一个网站的顶层（主）页。当你访问某个网站显示的默认页。

Host

请参阅Web主机。

Hosting

请参阅虚拟主机

Hotlink

请参阅超链接。

HTML (Hypertext Markup Language)

HTML就是web语言。HTML是一种用来定义内容，布局和网页文件格式的标签设置。 Web浏览器使用HTML标签来定义如何显示文本。

了解更多有关HTML的信息于我们HTML教程

HTML Document

在HTML编写的文件。

HTML DOM (HTML Document Object Model)

一个HTML文档编程接口。

了解更多有关HTML DOM的信息于我们HTML DOM教程

HTML Editor

一个软件程序，用于编辑HTML页面。有了一个HTML编辑器你可以像使用字处理器，向HTML文档添加元素如列表表格布局，字体大小，和颜色。正在编辑时它会在网页上显示（所见即所得）HTML编辑器将正在编辑的内容显示在页面。

HTML Form

用户输入的形式传递回服务器。

了解更多关于HTML表单的信息于我们的HTML教程

HTML Page

一个HTML文件

HTML Tags

代码以识别文档的的不同部分使网页浏览器会知道如何显示。

了解更多关于HTML标记的信息于我们的HTML教程

HTTP (Hyper Text Transfer Protocol)

通过Internet发送文本文件的规则标准设置。它需要一端HTTP客户端程序，并在另一端HTTP服务器程序。

HTTP Client

一种计算机程序，从Web服务器请求服务。

HTTP Server

一种计算机程序，从Web服务器提供服务。

HTTPS (Hyper Text Transfer Protocol Secure)

和HTTP相同但提供安全的互联网的通信的SSL。 （另请参阅SSL）

Hyperlink

网页中链接其它网页的文本串称为HYPERLINK。

Hypermedia

扩展超文本，图形和音频。

Hypertext

超文本是交叉链接的，以这样的方式，读者可以阅读相关文件通过点击一个高亮显示的单词或符号或其他文件文本。（另请参阅超链接）

IAB (Internet Architecture Board)

一个理事会为Internet标准决策（另请参阅w3c）。

IE (Internet Explorer)

请参阅Internet Explorer。

IETF (Internet Engineering Task Force)

一个侧重于解决技术问题，在互联网上对IAB分组。

IIS (Internet Information Server)

一个适用于Windows操作系统的Web服务器。由微软开发。

IMAP (Internet Message Access Protocol)

一个电子邮件服务器检索电子邮件标准通信协议。 IMAP是很像POP，但更先进的。

了解更多有关IMAP的信息于我们的TCP / IP教程

Indeo

由英特尔公司开发的计算机视频编解码器。

Internet

一个世界性的连接数以百万计的电脑的网络。 （请参阅也万维网）

Internet Browser

请参阅Web浏览器。

Internet Explorer

微软浏览器。最常用浏览器。

了解更多有关浏览器的信息在我们浏览器一节

Internet Server

请参见Web服务器

Intranet

私有（封闭）互联网内部的LAN（局域网）运行。

IP (Internet Protocol)

请参阅TCP / IP协议。

IP Address (Internet Protocol Address)

在互联网上的每一台计算机的一个独特的识别号码（如197.123.22.240）

IP Number (Internet Protocol Number)

一个IP地址。

IP Packet

请参阅TCP / IP包。

IRC (Internet Relay Chat)

互联网系统，使用户可以在网上讨论。

IRC Client

一个计算机程序，使用户能够连接到IRC。

IRC Server

Internet服务器专用的IRC连接服务任务。

ISAPI (Internet Server API)

Internet Information Server应用程序编程接口（API）（参阅IIS）。

ISDN (Integrated Services Digital Network)

使用数字传输的电信标准，支持通过普通电话线数据通信。

ISP (Internet Service Provider)

提供访问互联网和网站托管。

Java

由Sun开发的一种编程语言。大多用于编程Web服务器和Web小程序。

Java Applet

请参阅网页的Applet。

JavaScript

互联网上最流行的脚本语言由Netscape开发。

了解更多有关JavaScript的信息去我们的JavaScript教程。

JPEG (Joint Photographic Expert Group)

旨在促进JPG和JPEG图形格式存储压缩图像。

JPEG and JPG

图形格式存储压缩图像。

JScript

微软版本的JavaScript。

JSP (Java Server Pages)

一个基于Java技术允许在网页中插入服务器可执行的脚本。主要用于在Linux，Unix和Solaris平台。

K

千字节10K同样是十千字节..

KB

千字节10K同样是十千字节..

Keyword

在web方面：搜索引擎来搜索相关网络信息的一个词。 在数据库术语：一个字（或指数）用于识别数据库中的记录。

Kilobyte

1024字节。通常被称为1K，向下调整至1000个字节。

LAN (Local Area Network)

在局部地区（如建筑物内）的计算机之间的网络，通常是通过当地的电缆连接。请参阅广域网。

Link

和超链接相同。

Linux

开放源码计算机操作系统，基于UNIX的。主要用于服务器和Web服务器。

Mail

在网络方面：和电子邮件相同。

Mail Server

请参阅e - mail服务器。

MB

和兆字节相同。 10MB是10兆字节。

Megabyte

1024千字节。通常向下舍入到一百万字节。

Meta Data

说明其他数据的数据。（meta标签）。

Meta Search

搜索文件中的元数据的方法。

Meta Tags

标签插入到文档中描述的文件。

了解更多有关meta标签的信息去我们的HTML教程。

MIDI (Musical Instrument Digital Interface)

一个计算机和乐器之间的沟通的标准协议。

了解更多有关MIDI的信息去我们的媒体教程。

MIME (Multipurpose Internet Mail Extensions)

定义文档类型的Internet标准。 MIME类型的例子：文本/纯文本，文本/图像/ GIF，HTML，图像/ JPG。

了解更多有关MIME的信息去我们的媒体教程。

MIME Types

根据MIME文件类型定义。

Modem

硬件设备将计算机连接到电话网络，通常用于通过电话线连接到互联网。

Mosaic

第一个常用的Web浏览器。Mosaic是在1993年发布，并开始普及Web。

MOV

由苹果公司开发的计算机视频编解码器。 QuickTime多媒体文件的通用文件扩展名。

MP3 (MPEG-1 Audio Layer-3)

专为方便下载的Internet.c设计的一种音频压缩格式

MP3 File

文件中包含音频压缩MP3。最常见的音乐曲目。

MPEG (Moving Picture Expert Group)

一个ISO标准计算机的音频和视频编解码器。

MPG

MPEG文件的通用文件扩展名。

MS-DOS (Microsoft Disk Operating System)

一个通用的基于磁盘的计算机操作系统（操作系统）。最初由微软开发为IBM电脑，然后由微软开发的第一个版本的Windows的基础。

Multimedia

在网络方面：结合文字与图片，视频或声音的演示文稿。

MySQL

经常被用来在网络上免费的开源数据库软件。

NetBEUI (Net Bios Extended User Interface)

一个增强版的NetBIOS。

NetBIOS (Network Basic Input Output System)

一个应用程序编程接口（API），在局域网功能（LAN）上。用于DOS和Windows。

Navigate

在网络方面：和浏览相同。

Netscape

该公司的Netscape浏览器。是多年来最流行的浏览器。今天，带领着IE浏览器。

在我们的浏览器部分了解更多有关浏览器的信息

Newsgroup

一个在线讨论小组（新闻服务器上的部分），专用于感兴趣的某个主题。

News Reader

一个计算机程序，可以读取（和POST消息）从Internet新闻组。

News Server

Internet服务器专用的互联网新闻组服务的任务。

Node

在连接到互联网的网络方面：一台电脑，最经常被用来描述一个Web服务器。

Opera

来自公司的Opera的浏览器。

在我们的浏览器部分了解更多有关浏览器的信息

OS (Operating System)

管理软件的电脑基本操作。

Packet

请参阅参阅TCP / IP包。

Page Hits

网页由用户访问的次数数量。

Page Impressions

点击次数相同。

Page Views

点击次数相同。

PDF (Portable Document Format)

一个文档文件格式，由Adobe开发。最常用的文本文件。

Perl (Practical Extraction and Reporting Language)

一个Web服务器的脚本语言。最常用在UNIX服务器上。

PHP (PHP: Hypertext Preprocessor)

一个技术，允许在网页中插入服务器可执行脚本。多用在Unix，Linux和Solaris平台。

了解更多有关PHP的信息去我们的PHP教程。

Ping

一个方法用来检查两台计算机之间的沟通。一个"ping"发送到远程计算机，看它是否响应。

Platform

在网络方面：电脑的作业系统，如Windows，Linux或OS X

Plug-In

到另一个应用程序构建的应用程序。在网络方面：（或添加）内置Web浏览器来处理像电子邮件，声音，或电影文件的数据的一个特殊类型的程序。（另见ActiveX）

PNG (Portable Network Graphics)

图像文件存储格式，其目的是试图替代GIF和TIFF文件格式，同时增加一些GIF文件格式所不具备的特性

POP (Post Office Protocol)

一个电子邮件服务器检索电子邮件的标准通信协议。（另见IMAP）。

了解更多有关POP和IMAP的信息去我们的TCP / IP教程。

Port

一个标识一台计算机的IO（输入/输出）通道。在网络方面：一个标识Internet应用程序（Web服务器通常使用80端口）所使用的I / O通道。

Protocol

请参阅通信协议。

PPP (Point to Point Protocol)

一个用于两台计算机之间的直接连接的通信协议。

Proxy Server

Internet服务器致力于改善互联网性能。

QuickTime

由苹果公司创建多媒体文件的格式。

在我们的媒体教程可以进一步了解QuickTime

RAID (Redundant Array of Independent Disks)

一个更高的安全性，速度和性能相同的服务器连接多个磁盘标准。通常用于Web服务器上。

RDF (Resource Description Framework)

用于描述Web资源建设的框架的语言。

了解更多有关RDF的信息去我们的RDF教程。

Real Audio

一个常见的多媒体音频格式，由Real Networks公司创建。

了解真正的音频去我们的媒体教程

Real Video

一个常见的多媒体视频格式，由Real Networks公司创建。

了解真正的视频去我们的媒体教程

Redirect

在网络方面：行动时，网页自动转发（重定向）到另一个网页的用户。

RGB (Red Green Blue)

可以代表全彩光谱的三原色组合。

了解更多有关RGB的信息去我们的HTML教程。

Robot

请参阅网络Robot。

Router

一个硬件（或软件）系统（路线），指示数据在不同的计算机上网络传输。

Schema

参见XML Schema。

Script

脚本语言编写的语句的集合。

Scripting Language

在网络方面：一个简单的编程语言，可以通过Web浏览器或Web服务器执行。 参考JavaScript和VBScript。

Scripting

编写一个脚本。

Search Agent

和搜索引擎相同。

Search Engine

计算机程序用于搜索和目录（索引）数以百万计的网页在网页上提供的信息。常见的搜索引擎Google和AltaVista。

Semantic Web

一个网站的意义，在这个意义上，计算机程序可以了解足够的数据与处理数据。

Server

请参阅Web服务器。

Server Errors

参考Web服务器错误。

Shareware

软件，你可以尝试免费的，并支付一定的费用，继续合法使用。

Shockwave

在网页中嵌入多媒体内容由Adobe开发的一种格式（技术）。

SGML (Standard Generalized Markup Language)

用于标记语言的国际标准。 HTML和XML的基础。

SMIL (Synchronized Multimedia Integration Language)

一个W3C推荐语言创建的多媒体演示。

SMTP (Simple Mail Transfer Protocol)

一个发送电子邮件的计算机之间的标准通信协议。

了解更多有关SMTP的信息去我们的TCP/IP教程。

SOAP (Simple Object Access Protocol)

让应用程序进行通信相互使用XML的一个标准协议。

了解更多有关SOAP的信息去我们的SOAP教程。

Solaris

来自Sun的计算机操作系统。

SPAM

在网络方面：多个不受欢迎的邮件发送到新闻组或邮件列表的行动。

Spider

请参阅Web Spider。

Spoofing

网页或虚假引荐的电子邮件的寻址。像假地址发送电子邮件。

Spyware

计算机软件隐藏在一台计算机与使用计算机收集信息的目的。

SQL (Structured Query Language)

访问和操作数据库的一个ANSI标准的计算机语言。

了解更多有关SQL的信息去我们的SQL教程。

SQL Server

来自微软的数据库系统。主要用于在高流量的网站，在Windows平台上运行的网站。

SSI (Server Side Include)

HTML注释类型插入到网页指示Web服务器生成动态内容。最常见的用途是包含标准页眉或页脚的页面。

SSL (Secure Socket Layer)

软件安全和保护网站沟通，采用加密数据传输。

Static IP (address)

静态ip,对应于动态ip。

Streaming

这样一种方式，用户可以查看正在传输的文件，同时通过互联网发送视频和音频文件的方法。

Streaming Format

在互联网上使用的文件格式流媒体。（请参阅Windows Media，Real视频和QuickTime）。

SVG (Scalable Vector Graphics)

一个W3C推荐的语言，在XML中定义图形。

了解更多有关SVG的信息去我们的SVG教程。

Tag

在网络方面：书面通知或进入网页文件命令。 （参阅HTML标签）

TCP (Transmission Control Protocol)

请参阅TCP / IP协议。

TCP/IP (Transmission Control Protocol / Internet Protocol)

两台计算机之间的互联网通信协议的集合。 TCP协议是两台计算机之间的自由连接，而IP协议负责通过网络发送的数据包。

了解更多有关TCP/IP 的信息去我们的TCP/IP教程。

TCP/IP Address

请参阅IP地址。

TCP/IP Packet

一个"包"一个TCP / IP网络上传送的数据。 （通过互联网发送的数据被分成从40到32000字节长的小"包"）

Trojan Horse

计算机程序，隐藏在另一台计算机破坏程序软件或使用计算机收集信息的目的。

UDDI (Universal Description Discovery and Integration)

独立于平台的框架，用于描述服务，探索业务，并使用互联网的集成业务服务。

了解更多有关UDDI的信息去我们的WSDL教程。

Unix

计算机操作系统，由贝尔实验室开发。大多用于服务器和Web服务器。

UNZIP

要解压压缩文件。请参阅zip。

Upload

从本地计算机传输文件到远程计算机。在网络方面：从Web客户端传输到Web服务器的文件。（见下载）。

URI (Uniform Resource Identifier)

用来确定在互联网上的资源。 URL是一种类型的URI。

URL (Uniform Resource Locator)

Web地址。标准的办法来解决互联网上的网页文件（页）（如：http://www.w3cschool.cc/）

USENET

一个世界性的新闻系统，通过互联网访问。（参阅新闻组）

User Agent

和网页浏览器相同。

VB (Visual Basic)

请参阅Visual Basic。

VBScript

来自微软的脚本语言。 VBScript是ASP中的默认脚本语言。也可用于程序的Internet Explorer。

了解更多有关VBScript的信息去我们的VBScript教程。

Virus

和计算机病毒相同

Visit

在网络方面：对一个网站的访问。常用来形容一个Web站点的访问者的活动。

Visitor

在网络方面：Web站点的访问者。常用来形容一个人访问Web站点（观赏）。

Visual Basic

来自微软的编程语言

VPN (Virtual Private Network)

两个远程站点之间的专用网络，通过一个安全加密的虚拟互联网连接（隧道）。

VRML (Virtual Reality Modeling Language)

一种编程语言，允许被添加到HTML文档中的3D效果。

W3C (World Wide Web Consortium)

该组织用于管理标准的WWW。

了解更多有关W3C的信息去我们的W3C教程。

WAN (Wide Area Network)

在一个广泛的网络连接在一起的计算机，比局域网大，通常是通过电话线连接。另见局域网。

WAP (Wireless Application Protocol)

一个旧标准的数字移动电话等无线终端上的信息服务。

Web Address

和一个URL或URI相同。参考网址。

Web Applet

一个可以在网上下载，并在用户的计算机上运行的程序。最经常用Java编写的。

Web Client

一个软件程序，用于访问网页。有时和一个Web浏览器相同，但经常被用来作为一个广义的术语。

Web Browser

一个软件程序，用于显示网页。

了解更多有关浏览器的信息在我们浏览器一节

Web Document

一个文件格式在网上进行传播。最常见的网页文件是标记语言如HTML或XML格式。

Web Error

请参阅Web服务器错误。

Web Form

请参阅HTML表单。

Web Host

一个Web服务器，如公司或个人提供网站空间的"主机"Web服务。

Web Hosting

提供虚拟主机服务的动作。

Web Page

旨在通过Web分发的一份文件（通常是一个HTML文件）。

Web Log

请参阅Blog。

Web Robot

请参阅Web Spider。

Web Server

服务器是为一种计算机到其他计算机提供服务或信息。在网络方面：一个服务器提供Web内容的Web浏览器。

Web Server Error

从Web服务器，显示一个错误的讯息。最常见的Web服务器错误"404文件未找到"。

在我们的HTML教程了解更多有关Web服务器的错误消息

Web Services

软件组件和Web服务器上运行的应用程序。服务器到其他计算机上，浏览器或个人提供这些服务，使用标准的通讯协议。

Web Site

属于公司或个人的网页的集合的相关的网站。

Web Spider

一种计算机程序，搜索互联网网页。常见的web spiders是一个像谷歌搜索引擎索引的网页。web spiders也被称为网络机器人或漫游者。

Web Wanderer

请参阅Web Spider。

Wildcard

一个字符用来代替任何字符（S）。最常用的作为一个星号（*）的搜索工具。

Windows 2000, Windows NT, Windows 95/98, Windows XP

来自Microsoft的电脑操作系统。

Windows Media

音频和视频格式，由Microsoft在互联网上开发。（参阅ASF，ASX，WMA和WMF）。

在我们的媒体教程了解更多有关Windows Media的信息

WINZIP

一个压缩和解压文件的计算机程序。请参阅zip。

WMA

互联网的音频文件格式，由Microsoft开发。（请参阅WMV）。

在我们的媒体教程中学习更多的媒体格式。

WMV

互联网的视频文件格式，由Microsoft开发。（另见的WMA）。

在我们的媒体教程中学习更多的媒体格式。

WML (Wireless Markup Language)

旧标准用于无线终端，数字移动电话，从HTML继承，但基于XML的，远远比HTML更严格的信息服务。

WML Script

用于WML脚本语言（编程语言）。

Worm

计算机病毒，可以使自己的副本，并通过互联网传播到其他计算机。

WSDL (Web Services Description Language)

一个基于XML的语言，用于描述Web服务以及如何访问它们。

在我们的WSDL教程了解更多有关WSDL的消息

WWW (World Wide Web)

一个利用互联网进行交流的Web文档的计算机的全球网络。（请参阅互联网）

WWW Server

和Web Server相同。

WYSIWYG (What You See Is What You Get)

在Web方面：若要显示正在编辑的网页，以完全相同的方式显示在网页上。

XForms

替代版本的HTML表单，基于XML和XHTML。来自HTML表单的不同分离数据定义和数据显示。提供更丰富，更独立于设备的用户输入。

XHTML (Extensible Hypertext Markup Language)

以XML格式重新制定HTML。由W3C开发。

在我们的XHTML教程学习更多关于XHTML的信息

XPath

XPath是一种用于定义XML文档的部分（语言）的语法规则的设置。 XPath是一个W3C的XSL标准的重要组成部分。

在我们的XPath教程学习更多关于XPath的信息

XQuery

XQuery是一种用于从XML文档中提取信息（语言）的语法规则的设置。XQuery是有关XPath的基础。 XQuery是由W3C开发。

在我们的XQuery教程学习更多关于XQuery的信息

XML (Extensible Markup Language)

由W3C开发的Web文件，是专为SGML的一个简化版本。

在我们的XML教程学习更多关于XML的信息

XML Document

XML编写的一个文件。

在我们的XML教程学习更多关于XML的信息

XML DOM (XML Document Object Model)

一个由W3C制定的XML文档的编程接口。

在我们的XML DOM教程学习更多关于XML DOM的信息

XMLHttpRequest

编程接口（对象），所有现代Web浏览器对JavaScript都支持，使用网络浏览器和幕后（AJAX）的Web服务器之间交换数据。

在我们的AJAX教程学习更多关于XMLHttp的信息

XML Schema

XML Schema是以XML语言为基础的，它用于可替代DTD。XML schema文件描述了XML文档的结构。

在我们的XML Schema教程学习更多关于XML Schema的信息

XSD (XML Schema Definition)

和XML Schema大致相同。

XSL (Extensible Stylesheet Language)

一套由W3C开发的XML语言，包括XSLT，XSL - FO和XPath。

在我们的XSL教程学习更多关于XSL的信息

XSL-FO (XSL Formatting Objects)

用于格式化XML文档的XML语言。W3C XSL的一部分。

在我们的XSL-FO教程学习更多关于XSL-FO的信息

XSLT (XSL Transformations)

用于转换XML文档的XML语言。W3C XSL的一部分。

在我们的XSLT教程学习更多关于XSLT的信息

ZIP

一个计算机上的文件压缩格式。通常用于压缩文件。 使用象WinZip的计算机程序ZIP文件可以压缩（zipped）并解压缩。

SEO - 搜索引擎优化

您是否曾经使用过搜索引擎？你是否知道目前主要的搜索引擎有哪些？

本文列出了目前流行的搜索引擎列表。

SEO - 搜索引擎优化（Search Engine Optimization）

搜索引擎优化 (SEO) 是提高一个网站在搜索引擎中的排名（能见度）的过程。如果网站能够在搜索引擎中有良好的排名，有助于网站获得更多的流量。

SEO主要研究搜索引擎是工作的原理，是什么人搜索，输入什么搜索关键字）。优化一个网站，可能涉及内容的编辑，增加关键字的相关性。推广一个网站，可以增加网站的外链数量

搜索引擎优化需要修改一个网站的HTML源代码和网站内容。搜索引擎优化策略应在网站建设之前就纳入网站的发展，尤其是网站的菜单和导航结构。

笼统的说，所有使用作弊手段或可疑手段的，都可以称为黑帽SEO。比如说垃圾链接，隐藏网页，桥页，关键词堆砌等等。

帽seo就是作弊的意思，黑帽seo手法不符合主流搜索引擎发行方针规定。黑帽SEO获利主要的特点就是短平快，为了短期内的利益而采用的作弊方法。同时随时因为搜索引擎算法的改变而面临惩罚。

不论是白帽seo还是黑帽seo没有一个精准的定义。笼统来说所有使用作弊手段或一些可疑手段的都可称为黑帽SEO。例如隐藏网页，关键词堆砌，垃圾链接，桥页等等。

向搜索引擎提交网站

目前大多数搜索引擎提供了网站的提交路口，我们可以通过他们提供的入口提交站点，让搜索引擎能够及时抓取网站的数据。

- 360搜索引擎登录入口：http://info.so.360.cn/site_submit.html
- 百度搜索网站登录口：http://www.baidu.com/search/url_submit.html
- 百度单个网页提交入口：http://zhanzhang.baidu.com/sitesubmit
- Google网站登录口：http://www.google.com/addurl.html
- Google新闻网站内容：http://www.google.com/support/news_pub/bin/request.py?contact_type=suggest_content&hl=cn
- bing(必应)网页提交登录入口：http://www.bing.com/toolbox/submit-site-url
- 搜狗网站收录提交入口:http://www.sogou.com/feedback/urlfeedback.php

- SOSO搜搜网站收录提交入口:http://www.soso.com/help/usb/urlsubmit.shtml
- 雅虎中国网站登录口：http://search.help.cn.yahoo.com/h4_4.html
- 网易有道搜索引擎登录口：http://tellbot.youdao.com/report
- MSN必应网站登录口：http://cn.bing.com/docs/submit.aspx?FORM=WSDD2
- Alexa网站登录入口：http://www.alexa.com/help/webmasters
- TOM搜索网站登录口：http://search.tom.com/tools/weblog/log.php
- 铭万网B2B网址登陆口：http://search.b2b.cn/pageIncluded/AddPage.php
- 蚁搜搜索网站登录口：http://www.antso.com/apply.asp
- 快搜搜索网站登录口：http://www.kuaisou.com/main/inputweb.asp
- 汕头搜索登录口：http://www.stsou.com/join.asp
- 猎商登录口：http://www.lgoods.com/lg/lgss.htm
- 天网网站登陆口：http://home.tianwang.com/denglu.htm
- 速搜全球登陆口：http://www.suso.com.cn/suso/link.asp
- 快搜网站登陆口：http://www.kuaisou.com/main/inputweb.asp
- 搜猫搜索引擎登录入口：http://test.somao123.com/search/url_submit.php
- 泽许搜索网站登录入口：http://www.zxyt.cn/home/add/
- 简搜搜索引擎登陆口：http://www.jianso.com/add_site.html

通常情况下，您需要输入网站完整的URL，包括 http:// 前缀。

实例: http://www.w3cschool.cc/

你只需要向搜索引擎提交你的站点首页即可，搜索引擎会根据你的站点页面关联的链接找到其他页面。

你还可以在网页中添加描述和关键字，但不要期望这些影响您的网站排名。

搜索引擎索引会定期更新。如果你的站点有修改或者页面已删除，搜索引擎会定期进行修改与清理。

并不是所有的链接都会出现在搜索引擎当中。

W3C词汇和术语表

以A字母开头的词汇

英文	中文
abstract module	抽象模组
access	访问、存取
access control	存取控制
access control information	存取控制资讯
access mechanism	存取机制

access rights	存取权限
accessibility	无障碍性
accessibility information	无障碍网页资讯
accessibility problem	无障碍网页问题
accessible	无障碍的
accessible authoring practice	无障碍创作实践
acquired infoset	获取资讯集
ACSS (Audio cascading style sheets)	ACSS (音频阶层样式表)
activate	启用
active grammar	正在使用的文法
active perceivable unit	使用中的可视单元
activity	活动
actor	角色
adaptation	适应、调节
adaptation preferences	适配偏好设定
additional characters	附加字元
advisory board	谘询常委会
advisory committee	谘询委员会
age	寿命
agent	代理
aggregated authored units	集成创作单元
aggregation	集成
alert	警告 (信号)
alpha	alpha
alpha compaction	alpha 压缩、透明度
alpha separation	alpha 分色
alpha table	alpha 表
alternative information	替代资讯
amaya	amaya
ancestor	祖先、前辈结点
anchor	锚点

ancillary chunk	辅助数据块/区块
animation	动画
annotation	注释
anonymity	匿名性
antecedent	前提
apache	apache
applet	applet (应用小程序)
application	应用程式、应用
application personalization	应用程式个性化
application programming interface (API), conventional input/output/device API	应用程式编程界面 (API)，通用输入/输出/设备 API
arc	弧
architecture	架构
argument	参数
artifact	物件
ASCII art	ASCII 艺术
ASR	自动语音识别
assertion	断言
assistive technology	辅助技术
asynchronous	非同步的
asynchronous exchange	非同步交换
at user option	用户可选择的
atomic	原子的
atomic test	原子性测试
attribute	属性
attribute name	属性名称
attribute specifications	属性规范
attribute value	属性值
attribute, or CC/PP attribute	CC/PP 属性
attribute-list declarations	属性列表宣告
audio	音频、音讯
audio description	音频描述

audio track	音轨
audio-only presentation	纯音频演示
audit guard	稽核防护机制
auditory description	听觉描述
authentication	身份验证
author	作者
author styles	作者样式
authored unit	创作单元
authoring	创作
authoring tool	创作工具
authorization	授权
axis	座标轴

以B字母开头的词汇

英文	中文
back link	返回连结/链结
background image interference	背景图像干扰
backward compatible	向下兼容
base text	基础文本
baseline	基线
basic readability	基本可读性
binding	绑定、繫结
binding expression	绑定表达式
bit depth	色彩深度
black box	黑框
bopomofo	注音符号
bot	自动代理程序
bounding box	边界框、区域框
box	框
braille	点字
bridge	桥接器
browser	浏览器
button	按钮
byte	位元组
byte order	位元组顺序

以C字母开头的词汇

英文	中文
cache	快取
cacheable	可快取的
Candidate Recommendation (CR)	候选推荐标准 (CR)
capability	性能
captions	字幕
card	卡、卡片
cascading style sheets (CSS)	阶层样式表 (CSS)

catch element	catch 元素
CC/PP processor	CC/PP 处理器
CC/PP repository	CC/PP 资料储存库
CDATa sections	CDATa 区段
CERN	欧洲粒子物理研究所
certification	认证、凭证
chair	主持、(小组的) 主席
chairman	主席
channel	通道
character	字元
character data	字元资料
character data (CDATa)	字元资料 (CDATa)
character encoding	字元编码
character or expression depth	字元或表达式深度
character or expression height	字元或表达式高度
character or expression width	字元或表达式宽度
character reference	字元参考
check for	检查
child	子 (元素)
child	子 (节点)
choice	选择
choreography	(Webservice) 编排
chromaticity (CIE)	色度 (CIE)
chunk	数据块、区块
class	类别
class definition	类别定义
class description	类别描述
class name	类别名称
class of products	产品类别
click-stream	点击流、点选串流
client	客户端

collapse	折迭
collated text transcript	按序文字记录、逐字稿
colour type	色彩类型、颜色类型
comm	传讯、通讯
comments	注释、注解
complete	完备的
complex ruby markup	複杂旁注标记
compliance	一致性
component	组件
composite (verb)	组合、複合
computed expression	计算所得的表达式
concept	概念
condition	条件
conditional content	条件内容
conditional sections	条件区段
confidentiality	机密性
confidentiality	机密性、保密性
configuration	配置
configure, control	配置，控制
conformance	一致性
conformance clause	一致性条款
conformance level	一致性级别
conformance testing	一致性测试
conforming document	一致性文件
connection	(网络) 连接
consequent	结论、结果
consistent	一致的
constraint	限制
contained (element a is contained in B)	包含于 (元素A包含于元素B)
container (Constructor)	容器 (构造器)
containing document	容器文件
content	内容

content developer	内容开发者
content elements	内容元素
content generation	内容生成
content model	内容模型
content negotiation	内容协商
content provider	内容供应商
content selection	内容选择
content set	内容集
content token element	内容记号元素
context (of a given mathML expression)	(给定 mathML 表达式的) 上下文、取义
context node	上下文节点、取义节点
context position	上下文位置、取义位置
context size	上下文大小、取义大小
contradictory behaviors	矛盾行为
control	控制
control item	控制项目
convenience	便利
conversation	会话
conversion tool	转换工具
COO	首席营运总监、首席营运长
cookie	cookie
correct	正确的
credentials	凭证、凭据
critical chunk	关键数据块/区块
CSS (Cascading style sheets)	CSS (阶层样式表)
CSS W3C cascading style sheet specification	CSS W3C 阶层样式表规范
cyberspace	网络空间、网际空间
cyc	cyc (知识表示项目)

以D字母开头的词汇

英文	中文

daemon	独立后台程序
data category	资料类型
data element	资料元素
data model	资料模型
data resource	资料资源
data schema	资料纲目
data set	资料集
data structure	资料结构
data -valued property	资料类型属性
database	资料库
datastream	资料流
datatype	资料类型
datatype property	资料类型属性
date space	纲站编年区
decideable	可判定的
declaration	宣告
declared	已宣告的
decomposition	分解
deepest	最深的
default	预设
default namespace	预设命名空间
deferred request authentication	推迟请求验证
defining required attributes	定义必要属性
defining the type of attribute values	定义属性值类型
deflate	deflate (一种压缩演算法)
delivered image	传送完成的图像
delivery context	传送上下文、传送取义
delivery policy	传送政策
delivery unit	传送单位
deprecated	弃用
deprecated feature	已弃用的功能

depth	深度
dereference a URI	重新访问 URI
descendant	子节点
descendants	子节点
device	设备
device independent	与设备无关的
device-independence	设备无关
dialog	对话
digital rights management	数码/数位版权管理
digital signature	数码签署、数位签章
dimensions of variability (DoV)	变异维 (DoV)
direct sub-expression (of a mathML expression of)	(mathML 表达式的) 直接子表达式
directly contained (element a in B)	(a) 直接包含于 (B)
director	领导人、总监
discovery	探索
discovery service	探索服务
discretionary choices	任意选择
discretionary item	任意项目
document	文件
document character set	文件字元集
document content, structure, and presentation	文件内容，结构，表达
document entity	文件实体
document language	文件语言
document model	文件模型
document object model	文件物件模型
document object, document	文件物件，文件
document order	文件顺序
document profile	文件设置文件
document source, text source	文件来源，文本来源
document style semantics and specification language (DSSSL)	文件样式语义和规范语言 (DSSSL)

document tree	文件树
document type	文件类型
document type declaration	文件类型宣告
document type definition (DTD)	文件类型定义 (DTD)
documentation	参考文件
DOM (Document object model)	DOM (文件物件模型)
DOM (Document object model, see http://www.w3.org/DOM/)	DOM (文件物件模型)
DOM level 0	DOM level 0
domain	网域
domain name	网域名称
driver	驱动程式
DTD	DTD
DTD	文档类型定义 (DTD)
DTD-determined ID	DTD 确定 ID
DTMF (Dual tone multi-Frequency)	DTMF (双音多频)
dublin core	都柏林核心
dynamic content	动态内容
dynamic HTML (DHTML)	动态 HTML (DHTML)

以E字母开头的词汇

英文	中文
early normalization	提早规范化
ease of parsing and serializing:	剖析和序列化的简化
EBT (Electronic book technology)	电子图书技术
eCMAScript	eCMAScript
EDI (Electronic data interchange)	EDI (电子资料交换)
editing view	编辑阅览
electronic data interchange (EDI)	电子资料交换 (EDI)
element	元素
element content	元素内容

element name	元素名称
element type	元素类型
element type declaration	元素类型宣告
element, element type	元素，元素类型
elements	元素
embed	嵌入
embedded object	嵌入式物件
Embedded Web request	嵌入式 Web 请求
embellished operator	修饰操作符
empty	空
empty-element tag	空元素标籤
enabled element, disabled element	启用的元素，停用的元素
encryption	加密
end point	终点
end-tag	结束标籤
ending resource	结束资源
enquire	enquire (程式名)
entail	推导
entities	实体
entity	实体
entity reference	实体参考
enumerated attributes	枚举属性、列举属性
episode	情节
equable practice	等同的实践
equivalent	等价的
equivalent (for content)	(内容上) 等价
error	错误
error correction	错误修正
error recovery	错误修复
escape	转义
event	事件
events and scripting, event handler, event type	事件和脚本，事件处理器，事件类型

executable content	可执行内容
expanded name	扩展/扩充名称
expanded-name	扩展/扩充名称
explicit expiration time	显式过期时间
explicit user request	显式用户请求
Explicit Web request	显式 Web 请求
explicitly undefined behaviors	显式未定义行为
extended language	扩展语言
extended link	扩展连结/链结
extended links	扩展连结/链结
extending pre-defined elements	扩展预定义元素
extensible	可扩展的
extensible markup language (XML)	可扩展标记语言 (XML)
extensible style language (XSL)	可扩展样式语言 (XSL)
extension	扩充
extensional	外延的
external	外部的
external entity	外部实体
external markup declaration	外部标记宣告
externally-determined ID	外部可确定 ID

以F字母开头的词汇

英文	中文
facet	分面
facilities	设施
fatal error	致命错误
feature	特性、特徵、功能
fellow	研究员
fences	括号
Fla (Form interpretation algorithm)	Fla (表格解释演算法)
filter	过滤器

filtering	过滤
first node rule	首节点规则
first-hand	第一手的
flexible authoring	灵活创作
focus of attention	关注焦点
focus, content focus, user interface focus, current focus	焦点，内容焦点，用户界面焦点，目前焦点
following element	后继元素
font	字体
for compatibility	为了兼容性
for interoperability	为了互用性
form	表格
form control	表格控制
form item	表格项目
form item variable	表单项目变量
formal	正规
fragment identifier	碎片识别符
fragmentation	碎片
frame buffer	框缓冲区
fresh	未过期的
freshness lifetime	有效期
FTF	面对面
functional adaptation	功能适配
functional user experience	功能用户体验

以G字母开头的词汇

英文	中文
gamma	gamma
gateway	闸口
general entities	通用实体
generic identifier	通用识别符
GIF (Graphics interchange format)	GIF (图形交换格式)
GILC (Global internet liberty campaign)	GILC (全球互联网自由化运动)
glossary of terms for device independence	设备无关的术语表
glyph	字形
good practice	优秀实践
graphical	图形的
graphics	图形
greyscale	灰度、灰阶
group ruby	组旁注

以H字母开头的词汇

英文	中文
harmonized adaptation	协调适应
harmonized user experience	协调用户体验
height	高度
heuristic expiration time	启发式过期时间
highlight	突显
hint	提示
hiragana	平假名
host	主机
host	主办者
host language	主语言
host page	主网页、主页
HTML	HTML
HTML (Hypertext markup language)	HTML (超文本标记语言)
HTTP (Hypertext transfer protocol)	HTTP (超文本传输协议)
HTTP client	HTTP 客户端
HTTP gateway	HTTP 闸口
HTTP payload entity	HTTP 负载实体
HTTP proxy	HTTP 代理
HTTP representation	HTTP 表示
HTTP request	HTTP 请求
HTTP response	HTTP 回应
HTTP server	HTTP 伺服器
hybrid document	溷合文件
hyperlink	超连结/链结
hypermedia	超媒体
hypertext	超文本

以I字母开头的词汇

英文	中文
idempotent	恒等

identical	等同的
identified data	被识别的资料
identifier	识别符
ideograph	表意文字 (如方块字)
iff	当且仅当、若且唯若
image	图像、图形
image data	图像资料
image map	图像地图
implementation	实现、实作
implementation conformance statement (ICS)	实现一致性声明 (ICS)
implementation platform	实现平台
Implicit Web request	隐式 Web 请求
important	重要的
imports closure	汇入闭包
inbound	流入
inbound/outbound	流入/流出
include location	包含位置
include parent	包含父(项目)
included	包含的
included items	包含的项目
inclusion target	包含目标
inconsistent	不一致的
independent web	独立网页
index	索引
indexed-colour	索引色
indexical	索引的
indexing	索引
indirectly contained	间接包含
individual	个体
individual-valued property	个体价值属性
infer	推理

inform	通知
information resource	资讯资源
information set	资讯集
information space	资讯空间
informative	参考性的
informative text	参考文本
initial SOAP sender	初始 SOAP 发送者
input configuration	输入配置
input item	输入项目
input modalities	输入模态
INRIa (Institut national de recherche en infomatique et automatique)	INRIa (法国国家信息与自动化研究所)
instance	实例
instance data	实例资料
instance data node	实例资料节点
instance of	(类别的) 实例
instance of mathML	mathML 的实例
instantiate	示例
integrity	完整性
intensional	内涵的
interaction	交互、互动
interactive element, non-interactive element	交互式的元素，非交互式的元素
interlaced PNG image	交错的 PNG 图像
internal	内部的
internal entity	内部实体
internationalized resource identifier	国际化资源识别符
internet	互联网、网际网络
interoperability	互用性
interpretation	解释、理解
intranet	内联网
intrinsic dimensions	固有维度

inverse function	反函数
IP (Internet protocol)	IP (互联网协议)
IPR (Intellectual property rights)	IPR (知识产权)
IRC	IRC (互联网中继聊天)
IRI reference	IRI 参考
ISO (International standards organization)	ISO (国际标准化组织)
ISP (Internet service provider)	ISP (互联网服务供应商)

以J字母开头的词汇

英文	中文
java	java (程式语言)
jigsaw	jigsaw (伺服器)
JPEG (Joint photographic experts group)	JPEG (联合图像专家组): 一种图像编码格式
JSGF	JSGF (Java API 语音语法格式)

以K字母开头的词汇

英文	中文
kana	<日> 假名
kanji	<日> 日本汉字
katakana	<日> (日本字母) 片假名
keio university	庆应大学 (日本)
key	键、密钥、关键
key binding	密钥绑定
key location	密钥定位
key management	密钥管理
key name	密钥名称
key validation	密钥验证

以L字母开头的词汇

英文	中文

lambda expression	Lambda 表达式
language binding	语言绑定
language identifier	语言识别符
late normalization	推迟规范化
layout schema (plural: schemata)	佈局纲目
LCS (Laboratory for computer science)	LCS (计算机科学实验室)
LEAD (Live early adoption and demonstration)	LEAD (早期採用及示范政策)
level	级、层
lexical space	词法空间
libwww	libwww (WWW相关程序模块库)
line-mode	(命令) 行模式
line-mode browser	(命令) 行模式浏览器
linearized table	线性化表格
link	连结/链结
link text	连结/链结描述文字
linkbases	连结/链结库
linking element	连结/链结元素
list	列表
literal	字面
literal entity value	字面实体值
live	使用中的
local name	本地名称
local part	本地部分
local resource	本地资源
logic	逻辑
longfellow	longfellow (W3C 电话会议的 24 线连接器)
loose coupling	鬆散耦合
lossless compression	无损压缩
lossy compression	有损压缩
luminance	亮度

LZ77	LZ77 (数据压缩算法)

以M字母开头的词汇

英文	中文
machine understandable	机器可理解的
manageable service	可管理的服务
management	管理
management capability	管理能力
management interface	管理界面
management policy	管理政策
management semantics	管理中的语义
Manifestation	表徵
MARC record	MARC 记录 (机读目录记录)
markup	标记
markup declaration	标记宣告
markup language	标记语言
markup model	标记模型
match	匹配
mathematical markup language (MathML)	数学标记语言 (MathML)
mathML element	mathML 元素
mathML expression (within some valid mathML	mathML 表达式
may	可以
media type	媒体类型
member	会员
menu	选单
message	讯息
message correlation	讯息相关性
message exchange pattern (MEP)	讯息交换模式 (MEP)
message receiver	讯息接收者
message reliability	讯息可靠性

message sender	讯息发送方
message transport	讯息传输
meta -	元的 (前缀、表示一个事物应用于其自身)
metadata	元数据
metaphysical	形而上的
micropayments	微支付、小额付款
minimal constraint, principle of	最小约束原理
MIT (Massachusetts institute of technology)	MIT (麻省理工学院)
mixed content	溷合内容
mixed initiative	溷合式驱动
modality	模态
model binding expression	模型绑定表达式
model item	模型项目
model item property	模型项目属性
model theory	模型论
modularization	模块化
modularization model	模块化模型
module	模组
monoruby	单一旁注
monotonic	单调的
mosaic	mosaic (最早出现在 Internet 上的 Web 浏览器)
multi-purpose internet mail extensions (MIME)	多用途互联网邮件扩充 (MIME)
multiple authoring	多元化创作
must	必须
mystic	mystic (W3C 电话会议的 6 线连接器)

以N字母开头的词汇

英文	中文
name	名称、名字

named class	具名类别
namespace	命名空间
namespace document	名命空间文件
namespace name	命名空间名称
namespace prefix	命名空间前缀
namespace-valid	命名空间有效的
namespace-validating	命名空间验证
namespace-well-formed	命名空间良构的
natural language	自然语言
navigation	导览
navigation bars	导览栏
navigation mechanism	导览机制
NCSa (National center for supercomputing applications)	NCSa (美国国家超级应用计算中心)
negotiate content	协商内容
negotiation metadata	协商元数据
nelson, ted	nelson, ted
net	互联网 (Internet)
network byte order	网络位元组顺序
new	新的
neXT	neXT (公司名称)
NNTP (Network news transfer protocol)	NNTP (网络新闻传输协议)
node	节点
non-repudiation	不可否认性
non-variant content	无差异内容
none	无
nonmonotonic	非单调的
normative	规范性的
normative text	规范文本
normative, informative	规范性的，参考性的
notation declarations	符号宣告
notations	符号、记法

note	笔记

以O字母开头的词汇

英文	中文
object	物件
object property	物件属性
obligation	义务
obsolete feature	已淘汰的功能
occurs as attribute value	作为属性值出现
office	办事处
onLoad	onLoad
onRequest	onRequest
ontological	本体论的
ontology	本体论、本体
ontology document	本体文件
open source	开放源码
openMath	openMatch
operating environment	操作环境
operation	操作
operator, an mo element	操作符，一个 mo 元素
operator, content element	操作符 -> 操作符，内容元素
optional	可选择的
optional behaviors	可选择行为
optional features	可选择功能
orchestration	编排
origin server	源始伺服器
other	其他
otherwise	否则
outbound	流出
output modalities	输出模态
override	重载、改写
OWL class	OWL 类别
OWL Web Ontology Language Guide	OWL 网络本体语言指南

以P字母开头的词汇

英文	中文
packet	包、封包
page view	页面阅览
palette	调色板
parameter entities	参数实体
parameter entity	参数实体
parameter-entity references	参数实体参考
parent	父 (节点)
parent document type	父文件类型
parsed character data (PCDATa)	已剖析字元资料 (PCDATa)
parsed entity	已剖析实体
parsed entity's	已剖析实体的
parsing	解析
partial understanding	部分理解
partially selected	部份选择
participate	参与
pass extraction	阶段撷取
pass phrase key	通行密钥
path	路径
payload security	负载安全
perceivable unit	可感知单元
permission	许可
permission guard	许可保护机制
person or organization	个人或组织
personal digital assistant (PDa)	掌上电脑、电子手掌 (PDa)
PGP (Pretty good privacy)	PGP
physical transducer	物理的转换器
PICS (Platform form	PICS
pixel	像素
PKC (public key cryptography)	PKC (公钥密码学)

PKI (Public key infrastructure)	PKI (公钥基础建设)
placeholder	占位
plug-in	外挂
PNG (Portable network graphics)	PNG (可携式网络图形格式)
PNG datastream	PNG 资料流
PNG decoder	PNG 解码器
PNG editor	PNG 编辑器
PNG encoder	PNG 编码器
PNG file	PNG 档桉
PNG four-byte signed integer	PNG 四位元组符号整数
PNG four-byte unsigned integer	PNG 四位元组无符号整数
PNG image	PNG 图像
PNG signature	PNG 签署/签章
point	点
point of regard	注视点
pointer	指标
pointer part	指标部分
policy	政策
policy guard	政策保护机制
practice	实践
pre-defined function	预定义函数
preceding element	前序元素
preference	偏好设定
presentation elements	表达元素
presentation layout schema	表达佈局纲目
presentation markup	表达标记
presentation token element	表达标记元素
preserve	保持
principal	主实体
principal node type	首要节点类型
principle	原理

priority 1 (P1)	第 1 优先 (P1)
priority 2 (P2)	第 2 优先 (P2)
priority 3 (P3)	第 3 优先 (P3)
privacy	私隐
privacy policy	私隐政策
process	进程、处理
processing instructions	数据处理指令
profile	设置文件
profiling	设置文件
prompt	提示、提示输入
proof of possession (POP)	证明所有权 (POP)
properties, values, and defaults	属性，值和预设
property	属性
property definition	属性定义
Proposed Edited Recommendation	已修正的提议推荐标准
Proposed Recommendation (PR)	提议推荐标准 (PR)
proposition	命题
protection	保护
protocol	协议、协定
provider agent	供应商代理
provider entity	供应商实体
proximity position	近似位置
proxy	代理、代理伺服器
public identifier	公共识别符
publish	发表、发佈
publisher	发佈者
purpose	目的、意图

以Q字母开头的词汇

英文	中文
qualified name	限定名称
qualified names	限定名称
qualifier	限定符
quality assurance, Qa	品质保証
quality of service	服务品质、QoS

以R字母开头的词汇

英文	中文
RDF (Resource description framework)	RDF (资源描述架构)
RDF resource	RDF 资源
reader	读者
reading	读音、读法
REC	REC
receiver	接收者
recognize	识别
recommendation	推荐、推荐标准
reduced image	简化过的图像
reference architecture	参考架构
reference image	参考图像
reference in attribute value	属性值参考
reference in content	内容参考
reference in DTD	DTD 参考
reference in entity value	实体值参考
registry	注册、登记
reify	具体化
relation	关係
remote resource	远端资源
render	渲染、显示
rendered content	渲染过/显示的内容
rendered content, rendered text	渲染过/显示的内容，文字

rendering	渲染、显示
rendering preferences	渲染/显示的偏好设定
repair content, repair text	修复内容
replace	替换
replaced element	被替换的元素
replacement text	替换文字
repository	资料库、档桉库
representation	表示
request	请求、要求
requester agent	请求方代理
requester entity	请求方实体
Rescinded Recommendation	作废的推荐标准
reserved	保留的
resource	资源
resource error	资源错误
resource manifestation	资源表徵
response	响应、回应
restriction	约束
restriction, global	全域约束
restriction, local	本地约束
result infoset	结果资讯集
results verification	结果确认
reverse document order	文件逆序
RFC (Request for comments)	RFC (徵求意见)
RGB merging	RGB 合併
root	根
RPC (remote procedure call)	RPC (远端程序调用)
RSa	RSa (加密算法)
ruby text	旁注文字

以S字母开头的词汇

英文	中文
safe	安全的
safe interaction	安全交户
safe zone	安全区域
sample	样本
sample depth	採样深度
sample depth scaling	採样深度映射
satisfy	满足
scanline	扫描线
schema	纲目
schema (pl., schemata)	纲目
schema constraint	纲目约束
schema representation constraint	纲目表示约束
schema , RDF schema	纲目，RDF 纲目
schema -determined ID	由纲目决定的标识
scheme	scheme
scope of a declaration	宣告有效范围
screen magnifier	屏幕/萤幕放大器
screen reader	读屏器
scribe	会议记录员
script	脚本语言、脚本
secondary resource	次要资源
security	安全
security administration	安全管理
security architecture	安全架构
security auditing	安全审核
security domain	安全网域
security mechanism	安全机制
security model	安全模型
security policy	安全政策
security policy expression	安全政策表达式

security service	安全服务
selected	选择的
selected sub-expression (of an maction element)	(maction 元素中的) 被选子表达式
selection, current selection	选择，目前选择
semantic	语义的
semantic requirement	语义需求 (同"测试断言")
semantic web	语义网
semantically transparent	语义透明的
sender	发送者
separation of form from content	内容形式相分离
serial access, sequential navigation	循序访问，顺序导览
server	伺服器、伺务器端程式
server session	伺服器 session
service	服务
service description	服务描述
service interface	服务界面
service intermediary	服务中介
service provider	服务供应商
service provider (Data controller, legal entity)	服务供应商 (资料管理员，法人实体)
service requester	服务请求者
service role	服务角色
service semantics	服务语义
service-oriented architecture	导向服务架构
session	session
set	集
SGML (Standard generalized markup language)	SGML (标准通用标记语言)
shall	必须
should	应该
sibling	兄弟
simple link	简单连结/链结

simple links	简单连结/链结
simple ruby markup	简单旁注标记
single authoring	单一创作
site maps	网站地图
size and color of non-text content	非文本内容的大小和颜色
SMIL (Synchronized multimedia integration language)	SMIL (同步多媒体集成语言)
SOAP	简单对象访问协议
SOAP application	SOAP 应用软件
SOAP binding	SOAP 绑定
SOAP body	SOAP 主体
SOAP envelope	SOAP 信封
SOAP fault	SOAP 错误
SOAP feature	SOAP 功能
SOAP header	SOAP 标头
SOAP header block	SOAP 标头块
SOAP intermediary	SOAP 中介
SOAP message	SOAP 讯息
SOAP message exchange pattern (MEP)	SOAP 讯息交换模式 (MEP)
SOAP message path	SOAP 讯息路径
SOAP module	SOAP 模组
SOAP node	SOAP 节点
SOAP receiver	SOAP 接收者
SOAP role	SOAP 角色
SOAP sender	SOAP 发送者
sophia	sophia (地名: Sophia -Antipolis)
source document	来源文件
source image	来源图像
source infoset	来源资讯集
space-like (MathML expression)	类空 (MathML 表达式)
specification	规范
speech	语音

speech synthesis	语音合成
SRGS (Speech recognition grammar specification)	SRGS (语音识别语法规范)
SSML (Speech synthesis markup language)	SSML (语音合成标记语言)
stale	陈旧的、过时的
standard	标准
standard generalized markup language (SGML)	通用标记语言标准
start-tag	起始标籤
starting resource	起始资源
state	状态
statement	声明
strict conformance	严格一致
string identity matching	字元串匹配
string indexing	字元串索引
string-value	字元串值
structural markup	结构化标记
style sheet	样式表
style sheets	样式表
sub-expression (of a mathML expression)	(mathML 表达式的) 子表达式
subdialog	子对话
submission	提交的文档
subset language	子集语言
subsite	子站
suggested rendering rules for mathML presentation elements	推荐的 mathML 表达元素渲染/显示规则
supersite	父站
support, implement, conform	支持，实现，符合
supported	支持、支援
SVG (Scalable vector graphics)	SVG (可缩放向量图形)
synchronize	同步
synchronous	同步的
synthesis processor	合成处理器
system entity	系统实体

system identifier	系统识别符
sysWeb	sysWeb (W3C 系统网络组)

以T字母开头的词汇

英文	中文
tables of contents	目录
tabular information	表格式资讯
tag	标籤
TAG	技术架构组
tangle	tangle (程式名)
tapered prompts	渐缩式提示
TCP (Transmission control protocol)	TCP (传输控制协议)
team	团队
technical architecture group	技术架构组
technical report	技术报告
term taken verbatim from another source	自其他来源逐字沿用的词彙
test area	测试区
test assertion	测试断言
test case	测试桉例
test framework	测试框架
test purpose	测试目的
test requirement	测试需求、测试断言
test suite	测试集
testability	可测试性
TEX	TEX
text	文本
text content, non-text content	文本内容，非文本内容
text decoration	文本修饰
text transcript	文本抄本
text-To-Speech	文语转换
the empty string	空字元串

third-party	第三方
throw	抛出
time parameters	时间参数
TLS	传输层安全
tobin	tobin (人名: MauriceJ.Tobin)
token	标记
token element	标记元素
tokenized	标记化的
top-level element (of mathML)	(mathML 的) 最高层元素
top-level included items	最高层包含项目
topology	拓扑结构
tracing	跟踪
transaction	交易
transcript	抄本
transformation	转换、变换
traversal	遍历
triple	三元组
truecolour	全彩
trust service	信任服务
TTS	TTS (文语转换)
tunnel	通道
type	类型
typeface	字体

以U字母开头的词汇

英文	中文
UCS	通用字元集
UI or action binding expression	用户界面或动作绑定的表达式
ultimate SOAP receiver	最终 SOAP 接收者
unconditional conformance	无条件符合
uniform resource identifier	统一资源识别符

uniform resource identifier (URI)	统一资源识别符 (URI)
union	联合、合併
universe	全球的、通用的
unnamed class	未命名类别
unparsed entity	非剖析实体
unsafe interaction	不安全交互
unspecified	未指定的
upstream/downstream	上游/下游
URI	统一资源标识符
URI (Universal resource identifier)	URI (统一资源识别符)
URI aliases	URI 别名
URI collision	URI 冲突
URI ownership	URI 拥有权
URI persistence	URI 恆久性
URI reference	URI 参考
URIs	URIs
URL	统一资源定位器
URL (Uniform resource locator)	URL (统一资源定位器)
usage auditing	使用审核
usage scenario	使用场景
use	使用、採用
use case	用例
user	用户、使用者
user agent	用户代理，useragent
user agent (Ua)	用户代理，Ua
user agent default styles	用户代理预设样式
user agent profile	用户代理设置文件
user control of every user interface component	每个用户界面组建的用户控制
user experience	用户体验
user experience preferences	用户体验偏好设定
user interface, user interface control	用户界面，用户界面控制
user session	用户 session

user styles	用户样式
User-input Web request	用户输入式 Web 请求

以V字母开头的词汇

英文	中文
valid	有效的
valid mathML data	有效的 mathML 资料
valid style sheet	有效的样式表
validating processors	验证处理器
validation	验证
validation rule	验证规则
validation, validate, validating	验证，确认，校验
validator	验证器、校验器
validity constraint	有效性约束
value space	数值空间
variant	变体
variant content	可变内容
versioning	版本化
video	视频、视讯
view	阅览
view, viewport	阅览，阅览窗口
viola	viola 语言
virtual hypertext	虚拟超文本
visual track	视轨
visual-only presentation	纯视觉演示
visualText	可视化文本
vocabulary	词彙
voice	语音
voice browser	语音浏览器
voiceXML document	voiceXML 文件
voiceXML interpreter	voiceXML 解释/直译器
voiceXML interpreter context	voiceXML 解释/直译环境
VRML	虚拟现实建模语言
VRML (Virtual reality modeling language)	VRML (虚拟现实建模语言)

以W字母开头的词汇

英文	中文
W3C	W3C
W3C (World wide web consortium)	W3C (万维网联盟)
W3C recommendation	W3C 推荐标准
W3C Recommendation (REC)	W3C 推荐标准 (REC)
WAI (Web accessibility initiative)	WAI (无障碍网页倡议)
WAIS (Wide area information servers)	广域资讯服务系统
web	万维网
web agent	web 代理
web client	web 客户端
web collection	web 集合
web core	web 核心
web neighborhood	web 邻居
web page	网页
web page identifier	网页识别符
web periphery	web 外设
web request	web 请求
web request body	web 请求主体
web request header	web 请求标头
web resource	web 资源
web response	web 回应
web response body	web 回应主体
web response header	web 回应标头
web server	web 伺服器
web service	web 服务
web site	网站
web site publisher	网站发佈者
well-formed	良构的
well-formedness constraint	良构性约束
white point	白色点

width (of a box)	(文本或图形框的) 宽度
Working Draft (WD)	工作草案
Working Group Note	工作组笔记
world	世界、领域
world wide web	万维网
worldWideWeb (one word; no spaces)	浏览器
WWW	WWW (万维网)

以X字母开头的词汇

英文	中文
X	X
xanadu	xanadu
xForms model	xForms 模型
xForms processor	xForms 处理器
XLL (eXtensible linking language)	XLL (可扩展连结/链结语言)
XML (Extensible markup language)	XML (可扩展标记语言)
XML declaration	XML 宣告
XML document	XML 文件
XML name	XML 名称
XML namespace	XML 命名空间
XML processor	XML 处理器
xML-based format	基于 XML 的格式
xPointer processor	xPointer 处理器
XSL (Extensible style sheet language)	XSL (可扩展样式表语言)
XSL formatting objects (XSL FO)	XSL 格式化物件 (XSL FO)
XSL transformation (XSLT)	XSL 转换语言 (XSLT)

以Z字母开头的词汇

英文	中文
zakim	zakim (W3C 视频会议)
zlib	zlib 压缩格式

Web浏览器

浏览器 统计

浏览器的使用情况如何？

浏览器统计及发展趋势



统计数据是非常重要的信息。

从下面的统计（根据 W3CSchool 上近几年的日志文件），您可以看到本站用户的浏览器使用的长期趋势。

我们可以看到，Google Chrome、Firefox 和 Internet Explorer 是目前最常用的浏览器。

浏览器统计

2014	Chrome	Internet Explorer	Firefox	Safari	Opera
5 月	59.2 %	8.9 %	24.9 %	3.8 %	1.8 %
4 月	58.4 %	9.4 %	25.0 %	4.0 %	1.8 %
3 月	57.5 %	9.7 %	25.6 %	3.9 %	1.8 %
2 月	56.4 %	9.8 %	26.4 %	4.0 %	1.9 %
1 月	55.7 %	10.2 %	26.9 %	3.9 %	1.8 %

2013	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	55.8 %	9.0 %	26.8 %	3.8 %	1.9 %
11 月	54.8 %	10.5 %	26.8 %	4.0 %	1.8 %
10 月	54.1 %	11.7 %	27.2 %	3.8 %	1.7 %
9 月	53.2 %	12.1 %	27.8 %	3.9 %	1.7 %
8 月	52.9 %	11.8 %	28.2 %	3.9 %	1.8 %
7 月	52.8 %	11.8 %	28.9 %	3.6 %	1.6 %
6 月	52.1 %	12.0 %	28.9 %	3.9 %	1.7 %
5 月	52.9 %	12.6 %	27.7 %	4.0 %	1.6 %
4 月	52.7 %	12.7 %	27.9 %	4.0 %	1.7 %
3 月	51.7 %	13.0 %	28.5 %	4.1 %	1.8 %
2 月	50.0 %	13.5 %	29.6 %	4.1 %	1.8 %
1 月	48.4 %	14.3 %	30.2 %	4.2 %	1.9 %

2012	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	46.9 %	14.7 %	31.1 %	4.2 %	2.1 %
11 月	46.3 %	15.1 %	31.2 %	4.4 %	2.0 %
10 月	44.9 %	16.1 %	31.8 %	4.3 %	2.0 %
9 月	44.1 %	16.4 %	32.2 %	4.2 %	2.1 %
8 月	43.7 %	16.2 %	32.8 %	4.0 %	2.2 %
7 月	42.9 %	16.3 %	33.7 %	3.9 %	2.1 %
6 月	41.7 %	16.7 %	34.4 %	4.1 %	2.2 %
5 月	39.3 %	18.1 %	35.2 %	4.3 %	2.2 %
4 月	38.3 %	18.3 %	35.8 %	4.5 %	2.3 %
3 月	37.3 %	18.9 %	36.3 %	4.4 %	2.3 %
2 月	36.3 %	19.5 %	37.1 %	4.5 %	2.3 %
1 月	35.3 %	20.1 %	37.2 %	4.3 %	2.4 %

2011	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	34.6 %	20.2 %	37.7 %	4.2 %	2.5 %
11 月	33.4 %	21.2 %	38.1 %	4.2 %	2.4 %
10 月	32.3 %	21.7 %	38.7 %	4.2 %	2.4 %
9 月	30.5 %	22.9 %	39.7 %	4.0 %	2.2 %
8 月	30.3 %	22.4 %	40.6 %	3.8 %	2.3 %
7 月	29.4 %	22.0 %	42.0 %	3.6 %	2.4 %
6 月	27.9 %	23.2 %	42.2 %	3.7 %	2.4 %
5 月	25.9 %	24.9 %	42.4 %	4.0 %	2.4 %
4 月	25.6 %	24.3 %	42.9 %	4.1 %	2.6 %
3 月	25.0 %	25.8 %	42.2 %	4.0 %	2.5 %
2 月	24.1 %	26.5 %	42.4 %	4.1 %	2.5 %
1 月	23.8 %	26.6 %	42.8 %	4.0 %	2.5 %

2010	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	22.4 %	27.5 %	43.5 %	3.8 %	2.2 %
11 月	20.5 %	28.6 %	44.0 %	4.0 %	2.3 %
10 月	19.2 %	29.7 %	44.1 %	3.9 %	2.2 %
9 月	17.3 %	31.1 %	45.1 %	3.7 %	2.2 %
8 月	17.0 %	30.7 %	45.8 %	3.5 %	2.3 %
7 月	16.7 %	30.4 %	46.4 %	3.4 %	2.3 %
6 月	15.9 %	31.0 %	46.6 %	3.6 %	2.1 %
5 月	14.5 %	32.2 %	46.9 %	3.5 %	2.2 %
4 月	13.6 %	33.4 %	46.4 %	3.7 %	2.2 %
3 月	12.3 %	34.9 %	46.2 %	3.7 %	2.2 %
2 月	11.6 %	35.3 %	46.5 %	3.8 %	2.1 %
1 月	10.8 %	36.2 %	46.3 %	3.7 %	2.2 %

2009	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	9.8 %	37.2 %	46.4 %	3.6 %	2.3 %
11 月	8.5 %	37.7 %	47.0 %	3.8 %	2.3 %
10 月	8.0 %	37.5 %	47.5 %	3.8 %	2.3 %
9 月	7.1 %	39.6 %	46.6 %	3.6 %	2.2 %
8 月	7.0 %	39.3 %	47.4 %	3.3 %	2.1 %
7 月	6.5 %	39.4 %	47.9 %	3.3 %	2.1 %
6 月	6.0 %	40.7 %	47.3 %	3.1 %	2.1 %
5 月	5.5 %	41.0 %	47.7 %	3.0 %	2.2 %
4 月	4.9 %	42.1 %	47.1 %	3.0 %	2.2 %
3 月	4.2 %	43.3 %	46.5 %	3.1 %	2.3 %
2 月	4.0 %	43.6 %	46.4 %	3.0 %	2.2 %
1 月	3.9 %	44.8 %	45.5 %	3.0 %	2.3 %

2008	Chrome	Internet Explorer	Firefox	Safari	Opera
12 月	3.6 %	46.0 %	44.4 %	2.7 %	2.4 %
11 月	3.1 %	47.0 %	44.2 %	2.7 %	2.3 %
10 月	3.0 %	47.4 %	44.0 %	2.8 %	2.2 %
9 月	3.1 %	49.0 %	42.6 %	2.7 %	2.0 %
8 月	51.0 %	43.7 %	2.6 %	2.1 %	
7 月	52.4 %	42.6 %	2.5 %	1.9 %	
6 月	54.2 %	41.0 %	2.6 %	1.7 %	
5 月	54.4 %	39.8 %	2.4 %	1.5 %	
4 月	54.8 %	39.1 %	2.2 %	1.4 %	
3 月	53.9 %	37.0 %	2.1 %	1.4 %	
2 月	54.7 %	36.5 %	2.0 %	1.4 %	
1 月	54.7 %	36.4 %	1.9 %	1.4 %	

2007	Mozilla	Internet Explorer	Firefox	Safari	Opera
11 月	1.2 %	56.0 %	36.3 %	1.8 %	1.6 %
9 月	1.2 %	57.2 %	35.4 %	1.6 %	1.5 %
7 月	1.4 %	58.5 %	34.5 %	1.5 %	1.9 %
5 月	1.3 %	58.9 %	33.7 %	1.5 %	1.7 %
3 月	1.3 %	58.7 %	31.8 %	1.6 %	1.6 %
1 月	1.5 %	58.6 %	31.0 %	1.7 %	1.5 %

2006	Mozilla	Internet Explorer	Firefox	Netscape	Opera
11 月	2.5 %	60.6 %	29.9 %	0.2 %	1.5 %
9 月	2.3 %	62.1 %	27.3 %	0.4 %	1.6 %
7 月	2.3 %	62.4 %	25.5 %	0.4 %	1.4 %
5 月	2.3 %	63.0 %	25.7 %	0.3 %	1.5 %
3 月	2.4 %	64.7 %	24.5 %	0.5 %	1.5 %
1 月	3.1 %	66.0 %	25.0 %	0.5 %	1.6 %

2005	Mozilla	Internet Explorer	Firefox	Netscape	Opera
11 月	2.8 %	68.9 %	23.6 %	0.4 %	1.5 %
9 月	2.5 %	75.5 %	18.0 %	0.4 %	1.2 %
7 月	2.6 %	73.8 %	19.8 %	0.5 %	1.2 %
5 月	3.1 %	71.6 %	21.0 %	0.7 %	1.3 %
3 月	3.3 %	72.5 %	18.9 %	1.0 %	1.9 %
1 月	3.4 %	74.5 %	16.6 %	1.1 %	1.9 %

2004	Mozilla	Internet Explorer	Netscape	Opera
11 月	16.5 %	76.2 %	1.7 %	1.6 %
9 月	13.7 %	79.0 %	2.0 %	1.7 %
7 月	12.6 %	80.4 %	2.2 %	1.6 %
5 月	9.5 %	81.9 %	2.4 %	1.6 %
3 月	7.9 %	82.8 %	2.8 %	1.4 %
1 月	5.5 %	84.7 %	2.4 %	1.5 %

2003	Mozilla	Internet Explorer	Netscape	Opera
11 月	7.2 %	84.9 %	2.6 %	1.9 %
9 月	6.2 %	86.6 %	2.7 %	1.8 %
7 月	5.7 %	87.2 %	2.7 %	1.7 %
5 月	4.6 %	87.7 %	3.3 %	1.4 %
3 月	4.2 %	88.0 %	3.4 %	1.2 %
1 月	4.0 %	84.6 %	4.0 %	

2002	AOL	Internet Explorer	Netscape
11 月	5.2 %	83.4 %	8.0 %
9 月	4.5 %	83.5 %	8.0 %
7 月	3.5 %	84.5 %	7.3 %
5 月	2.8 %	86.7 %	7.3 %
3 月	3.0 %	86.1 %	7.7 %
1 月	2.8 %	85.8 %	7.9 %

Internet Explorer	Microsoft Internet Explorer
Firefox	Mozilla Firefox（2005 年之前标识为 Mozilla）
Chrome	Google Chrome
Mozilla	Mozilla Suite（Gecko, Netscape）
Safari	Safari（与 Konqueror 在 2007 年之前都标识为 Mozilla）
Opera	Opera（自 2011 年后，Opera 包含了 Opera Mini）
Netscape	Netscape Navigator（2006 年之后标识为 Mozilla）
AOL	America Online（基于 Internet Explorer 和 Mozilla）

以上未列出低于 0.5% 占比的浏览器。

统计数据可能会误导

作为一个一名 web 开发人员，您不能只依赖于上面的统计数据。统计数据可能会误导。

注意：W3CSchool 网站的用户群体主要是广大的 web 技术爱好者。他们一般会寻求一种更适用的替代浏览器进行安装使用。普通用户则会更倾向于直接使用系统预装的浏览器。

提示：全球平均水平可能与您的网站无关。不同的网站吸引不同的用户。一些网站吸引的是使用专业硬件的专业开发人员，而另外一些网站吸引的则是使用旧电脑的业余爱好者。

总之，我们从 W3CSchool 日志文件中所收集来的统计数据，多年来浏览最新的日志文件分析，清楚地表明了长期以来浏览器的使用趋势。

引用

_"简单纯粹的真理很少是纯粹的，也从来不会是简单的。" _Oscar Wilde

_"首先必须去了解事实，然后您才有可能在悠闲的时候对这些事实进行扭曲。" _Mark Twain

其他的统计数据

操作系统统计

屏幕分辨率统计

操作系统（OS）平台 统计

操作系统的使用情况如何？

操作系统（OS）平台统计及发展趋势



统计数据是非常重要的信息。

从下面的统计（根据 W3CSchool 上近几年的日志文件），您可以看到长期以来本站用户操作系统的使用趋势。

操作系统平台（OS）统计

2014	Win8	Win7	Vista	NT*	WinXP	Linux	Mac	Mobile
5 月	16.6%	55.2%	1.2%	0.2%	7.3%	5.1%	10.0%	4.2%
4 月	15.8%	55.4%	1.2%	0.2%	8.0%	5.0%	10.3%	4.0%
3 月	15.0%	55.1%	1.3%	0.2%	9.4%	4.9%	9.9%	4.0%
2 月	14.2%	55.0%	1.4%	0.3%	10.1%	5.0%	10.0%	4.0%
1 月	13.4%	55.3%	1.5%	0.3%	11.0%	4.9%	9.6%	4.0%

2013	Win8	Win7	Vista	NT*	WinXP	Linux	Mac	Mobile
12 月	10.0%	55.9%	1.5%	3.1%	11.6%	4.8%	9.2%	3.8%
11 月	8.6%	56.4%	1.6%	3.7%	11.7%	4.8%	9.6%	3.7%
10 月	9.9%	56.7%	1.6%	1.4%	12.4%	4.9%	9.6%	3.3%
9 月	10.2%	56.8%	1.6%	0.4%	13.5%	4.8%	9.3%	3.3%
8 月	9.6%	55.9%	1.7%	0.4%	14.7%	5.0%	9.2%	3.4%
7 月	9.0%	56.2%	1.8%	0.4%	15.8%	4.9%	8.7%	3.2%
6 月	8.6%	56.3%	2.0%	0.4%	15.4%	4.9%	9.1%	3.2%
5 月	7.9%	56.4%	2.1%	0.4%	15.7%	4.9%	9.7%	2.6%
4 月	7.3%	56.4%	2.2%	0.4%	16.4%	4.8%	9.7%	2.2%
3 月	6.7%	55.9%	2.4%	0.4%	17.6%	4.7%	9.5%	2.3%
2 月	5.7%	55.3%	2.4%	0.4%	19.1%	4.8%	9.6%	2.2%
1 月	4.8%	55.3%	2.6%	0.5%	19.9%	4.8%	9.3%	2.2%

2012	Win8	Win7	Vista	NT*	WinXP	Linux	Mac	Mobile
12 月	2.5%	55.6%	2.8%	1.8%	21.1%	4.7%	8.7%	2.2%
11 月	56.5%	2.9%	3.0%	20.8%	4.8%	9.4%	2.0%	
10 月	56.8%	3.0%	1.8%	22.1%	4.8%	9.2%	1.8%	
9 月	55.7%	3.1%	1.5%	23.6%	4.7%	8.9%	1.8%	
8 月	54.5%	3.2%	1.3%	24.8%	5.0%	8.7%	1.8%	
7 月	53.8%	3.4%	1.2%	26.1%	4.9%	8.2%	1.7%	
6 月	53.2%	3.7%	1.1%	26.2%	5.0%	8.6%	1.6%	
5 月	52.3%	3.9%	1.1%	26.8%	4.9%	9.0%	1.6%	
4 月	51.3%	4.2%	1.0%	27.3%	4.9%	9.3%	1.5%	
3 月	49.9%	4.3%	1.0%	28.9%	4.9%	8.9%	1.4%	
2 月	48.7%	4.5%	0.8%	30.0%	5.0%	9.1%	1.3%	
1 月	47.1%	4.8%	0.9%	31.4%	4.9%	9.0%	1.3%	

2011	Win7	Vista	Win2003	WinXP	Linux	Mac	Mobile
12 月	46.1%	5.0%	0.7%	32.6%	4.9%	8.5%	1.2%
11 月	45.5%	5.2%	0.7%	32.8%	5.1%	8.8%	1.0%
10 月	44.7%	5.5%	0.7%	33.4%	5.0%	8.9%	1.0%
9 月	42.2%	5.6%	0.8%	36.2%	5.1%	8.6%	0.9%
8 月	40.4%	5.9%	0.8%	38.0%	5.2%	8.2%	0.9%
7 月	39.1%	6.3%	0.9%	39.1%	5.3%	7.8%	1.0%
6 月	37.8%	6.7%	0.9%	39.7%	5.2%	8.1%	0.9%
5 月	36.5%	7.1%	0.9%	40.7%	5.1%	8.3%	0.8%
4 月	35.9%	7.6%	0.9%	40.9%	5.1%	8.3%	0.8%
3 月	34.1%	7.9%	0.9%	42.9%	5.1%	8.0%	0.7%
2 月	32.2%	8.3%	1.0%	44.2%	5.1%	8.1%	0.7%
1 月	31.1%	8.6%	1.0%	45.3%	5.0%	7.8%	0.7%

2010	Win7	Vista	Win2003	WinXP	W2000	Linux	Mac
12 月	29.1%	8.9%	1.1%	47.2%	0.2%	5.0%	7.3%
11 月	28.5%	9.5%	1.1%	47.0%	0.2%	5.0%	7.7%
10 月	26.8%	9.9%	1.1%	48.9%	0.3%	4.7%	7.6%
9 月	24.3%	10.0%	1.1%	51.7%	0.3%	4.6%	7.2%
8 月	22.3%	10.5%	1.3%	53.1%	0.4%	4.9%	6.7%
7 月	20.6%	10.9%	1.3%	54.6%	0.4%	4.8%	6.5%
6 月	19.8%	11.7%	1.3%	54.6%	0.4%	4.8%	6.8%
5 月	18.9%	12.4%	1.3%	55.3%	0.4%	4.5%	6.7%
4 月	16.7%	13.2%	1.3%	56.1%	0.5%	4.5%	7.1%
3 月	14.7%	13.7%	1.4%	57.8%	0.5%	4.5%	6.9%
2 月	13.0%	14.4%	1.4%	58.4%	0.6%	4.6%	7.1%
1 月	11.3%	15.4%	1.4%	59.4%	0.6%	4.6%	6.8%

2009	Win7	Vista	Win2003	WinXP	W2000	Linux	Mac
12 月	9.0%	16.0%	1.4%	61.6%	0.6%	4.5%	6.5%
11 月	6.7%	17.5%	1.4%	62.2%	0.7%	4.3%	6.7%
10 月	4.4%	18.6%	1.5%	63.3%	0.7%	4.2%	6.8%
9 月	3.2%	18.3%	1.5%	65.2%	0.8%	4.1%	6.5%
8 月	2.5%	18.1%	1.6%	66.2%	0.9%	4.2%	6.1%
7 月	1.9%	17.7%	1.7%	67.1%	1.0%	4.3%	6.0%
6 月	1.6%	18.3%	1.7%	66.9%	1.0%	4.2%	5.9%
5 月	1.1%	18.4%	1.7%	67.2%	1.1%	4.1%	6.1%
4 月	0.7%	17.9%	1.7%	68.0%	1.2%	4.0%	6.1%
3 月	0.5%	17.3%	1.7%	68.9%	1.3%	4.0%	5.9%
2 月	0.4%	17.2%	1.6%	69.0%	1.4%	4.0%	6.0%
1 月	0.2%	16.5%	1.6%	69.8%	1.6%	3.9%	5.8%

2008	Vista	W2003	WinXP	W2000	Win98	Linux	Mac
12 月	15.6%	1.7%	71.4%	1.7%	0.1%	3.8%	5.3%
11 月	15.1%	1.6%	72.0%	1.8%	0.1%	3.8%	5.3%
10 月	14.4%	1.7%	72.2%	1.9%	0.2%	3.8%	5.5%
9 月	13.2%	1.8%	73.3%	2.2%	0.2%	3.8%	5.2%
8 月	12.5%	1.9%	73.9%	2.4%	0.2%	3.9%	4.9%
7 月	11.5%	2.0%	74.7%	2.6%	0.2%	3.9%	4.8%
6 月	10.0%	1.9%	74.6%	2.6%	0.2%	3.7%	4.8%
5 月	9.3%	1.8%	74.0%	2.9%	0.3%	3.6%	4.7%
4 月	8.8%	1.9%	73.3%	3.3%	0.5%	3.7%	4.6%
3 月	8.5%	1.9%	72.7%	3.7%	0.6%	3.9%	4.4%
2 月	7.8%	1.8%	72.4%	4.0%	0.8%	3.8%	4.3%
1 月	7.3%	1.9%	73.6%	4.0%	0.8%	3.6%	4.4%

2007	Vista	W2003	WinXP	W2000	Win98	Linux	Mac
11 月	6.3%	2.0%	73.8%	5.1%	1.0%	3.3%	3.9%
9 月	4.5%	2.0%	74.3%	5.4%	0.9%	3.4%	3.9%
7 月	3.6%	2.0%	74.6%	6.0%	0.9%	3.4%	4.0%
5 月	2.8%	1.9%	75.0%	6.5%	0.9%	3.4%	3.9%
3 月	1.9%	1.9%	76.0%	7.2%	0.9%	3.4%	3.8%
1 月	0.6%	1.9%	76.1%	7.7%	1.0%	3.6%	3.8%

2006	Win2003	WinXP	W2000	Win98	WinNT	Linux	Mac
11 月	1.9%	74.9%	8.0%	1.0%	0.3%	3.5%	3.6%
9 月	2.0%	74.6%	9.2%	1.4%	0.3%	3.5%	3.6%
7 月	2.0%	74.3%	10.1%	1.5%	0.3%	3.4%	3.6%
5 月	2.0%	74.2%	10.7%	1.6%	0.2%	3.4%	3.6%
3 月	1.8%	72.9%	11.9%	2.0%	0.3%	3.4%	3.5%
1 月	1.7%	72.3%	13.1%	2.4%	0.3%	3.3%	3.5%

2005	Win2003	WinXP	W2000	Win98	WinNT	Linux	Mac
11 月	1.7%	71.0%	14.6%	2.7%	0.4%	3.3%	3.3%
9 月	1.7%	69.2%	15.8%	3.2%	0.5%	3.3%	3.1%
7 月	1.6%	65.3%	17.7%	3.9%	0.6%	3.5%	3.0%
5 月	1.4%	64.5%	19.4%	3.9%	0.8%	3.3%	2.9%
3 月	1.4%	63.1%	20.2%	4.7%	0.9%	3.2%	3.0%
1 月	1.2%	61.3%	21.6%	5.3%	1.0%	3.2%	2.8%

2004	WinXP	W2000	Win98	WinNT	Win95	Linux	Mac
11 月	59.1%	23.7%	5.6%	1.2%	0.1%	3.1%	2.7%
9 月	55.9%	26.2%	6.4%	1.5%	0.2%	3.1%	2.6%
7 月	52.5%	28.4%	7.5%	1.9%	0.2%	3.1%	2.4%
5 月	51.0%	29.6%	8.2%	2.0%	0.3%	2.9%	2.5%
3 月	48.0%	31.1%	9.4%	2.4%	0.4%	2.6%	2.4%
1 月	44.1%	33.6%	10.4%	3.0%	0.4%	2.7%	2.4%

2003	WinXP	W2000	Win98	WinNT	Win95	Linux	Mac
11 月	42.6%	36.3%	10.9%	3.5%	0.4%	2.6%	2.2%
9 月	38.0%	37.9%	12.1%	4.1%	0.5%	2.4%	2.0%
7 月	33.9%	40.6%	12.6%	5.3%	0.6%	2.3%	1.9%
5 月	31.4%	41.0%	13.9%	5.8%	0.7%	2.2%	1.8%
3 月	29.1%	41.9%	14.8%	6.6%	0.8%	2.2%	1.8%

以上未列出低于 0.5% 占比的操作系统（OS）平台。

- NT 包含了所有的 Windows Server 操作系统（比如 Windows 2000、Windows Server 2003 和 2008）

统计数据可能会误导

作为一个一名 web 开发人员，您不能只依赖于上面的统计数据。统计数据可能会误导。

注意：W3CSchool 网站的用户群体主要是广大的 web 技术爱好者。他们一般会寻求一种更适用的替代浏览器进行安装使用。普通用户则会更倾向于直接使用系统预装的浏览器。

提示：全球平均水平可能与您的网站无关。不同的网站吸引不同的用户。一些网站吸引的是使用专业硬件的专业开发人员，而另外一些网站吸引的则是使用旧电脑的业余爱好者。

总之，我们从 W3CSchool 日志文件中所收集来的多年的统计数据，清楚地表明了长期以来操作系统的使用趋势。

引用

_"世界上有三种谎言：谎言、该死的谎言，以及统计数据。" _Benjamin Disraeli

其他的统计数据

浏览器统计

屏幕分辨率统计

屏幕分辨率 统计

屏幕分辨率及颜色深度的发展趋势如何？

Web 统计及发展趋势



统计数据是非常重要的信息。

W3CSchool 网站的用户群体主要是广大的 web 技术爱好者。这表明，下面的统计数据并非是 100% 针对普通互联网用户。针对普通的互联网用户统计出来的屏幕分辨率值可能会更低一些。

从下面的统计（根据 W3CSchool 上近几年的日志文件），您可以看到长期以来本站用户显示器的使用趋势。

总之，我们从 W3CSchool 日志文件中所收集来的多年的统计数据，清楚地表明了长期以来屏幕分辨率及颜色深度的发展趋势。

屏幕分辨率统计

目前，大多数用户的屏幕分辨率等于或大于 1024x768 像素：

日期	更高	1920x1080	1366x768	1280x1024	1280x800	1024x768	8
2014 年 1 月	34%	13%	31%	8%	7%	6%	0
2013 年 1 月	36%	11%	25%	10%	8%	9%	0
2012 年 1 月	35%	8%	19%	12%	11%	13%	1
2011 年 1 月	50%	6%	15%	14%	14%	0%	1

2010 年 1 月	39%	2%	18%	17%	20%	1%	3
2009 年 1 月	57%	36%	4%	3%			
2008 年 1 月	38%	48%	8%	6%			
2007 年 1 月	26%	54%	14%	6%			
2006 年 1 月	17%	57%	20%	6%			
2005 年 1 月	12%	53%	30%	5%			
2004 年 1 月	10%	47%	37%	6%			
2003 年 1 月	6%	40%	47%	7%			
2002 年 1 月	6%	34%	52%	8%			
2001 年 1 月	5%	29%	55%	11%			
2000 年 1 月	4%	25%	56%	15%			

颜色深度统计

今天，大多数计算机使用 24 位或 32 位硬件来显示 16,777,216 种不同的颜色。

较旧的计算机使用 16 位显卡，它最多能显示 65,536 种不同的颜色。

很旧的计算机使用 8 位显卡，它最多能显示 256 中不同的颜色。

日期	16,777,216	65,536	256
2014 年 1 月	98.5%	1%	0.5%
2013 年 1 月	98%	1.5%	0.5%
2012 年 1 月	98%	2%	0%
2011 年 1 月	97%	3%	0%
2010 年 1 月	97%	3%	0%
2009 年 1 月	95%	4%	1%
2008 年 1 月	90%	8%	2%
2007 年 1 月	86%	11%	2%
2006 年 1 月	81%	16%	3%
2005 年 1 月	72%	25%	3%
2004 年 1 月	65%	31%	4%
2003 年 1 月	51%	44%	5%
2002 年 1 月	43%	50%	7%
2001 年 1 月	37%	55%	8%
2000 年 1 月	34%	54%	12%

阅读更多关于颜色的显示信息

其他的统计数据

浏览器统计

操作系统统计

移动设备 统计

移动设备的使用情况如何？

移动设备统计及发展趋势



移动设备是一种小型计算设备。

在苹果的 iPhone 手机发布后，移动设备的使用量一直稳步增长。

今天的智能手机有大触摸屏、漂亮的用户界面，并且是高度优化的网页浏览。

从下面的统计（根据 W3CSchool 上的日志文件），您可以看到本站用户移动设备的使用趋势。

移动设备统计

下表是 操作系统统计 中关于移动设备使用情况的细节：

2014	总计	iOS *	Android（安卓）	其他
5 月	4.16 %	1.27 %	2.14 %	0.75 %
4 月	4.03 %	1.20 %	2.10 %	0.73 %
3 月	4.00 %	1.32 %	2.01 %	0.67 %
2 月	4.00 %	1.21 %	2.01 %	0.78 %
1 月	4.00 %	1.23 %	1.93 %	0.84 %

2013	总计	iOS *	Android（安卓）	其他
12 月	3.80 %	1.13 %	1.76 %	0.91 %
11 月	3.68 %	1.20 %	1.67 %	0.81 %
10 月	3.29 %	1.12 %	1.44 %	0.73 %
9 月	3.26 %	0.95 %	1.55 %	0.76 %
8 月	3.38 %	1.21 %	1.43 %	0.74 %
7 月	3.16 %	1.03 %	1.37 %	0.76 %
6 月	3.17 %	1.15 %	1.32 %	0.70 %
5 月	2.64 %	1.12 %	1.04 %	0.48 %
4 月	2.24 %	1.06 %	0.97 %	0.21 %
3 月	2.26 %	1.11 %	0.96 %	0.19 %
2 月	2.17 %	1.06 %	0.91 %	0.20 %
1 月	2.18 %	1.07 %	0.90 %	0.21 %

2012	总计	iOS *	Android（安卓）	其他
12 月	2.19 %	1.05 %	0.90 %	0.24 %
11 月	1.97 %	1.00 %	0.77 %	0.20 %
10 月	1.84 %	0.94 %	0.70 %	0.20 %
9 月	1.78 %	0.89 %	0.68 %	0.21 %
8 月	1.79 %	0.91 %	0.66 %	0.22 %
7 月	1.71 %	0.90 %	0.59 %	0.22 %
6 月	1.63 %	0.90 %	0.52 %	0.21 %
5 月	1.57 %	0.88 %	0.50 %	0.19 %
4 月	1.47 %	0.83 %	0.45 %	0.19 %
3 月	1.40 %	0.78 %	0.44 %	0.18 %
2 月	1.27 %	0.71 %	0.39 %	0.17 %
1 月	1.25 %	0.70 %	0.38 %	0.17 %

2011	总计	iPhone	iPad	iPod	Android（安卓）	其他
12 月	1.17 %	0.20 %	0.39 %	0.04 %	0.35 %	0.19 %
11 月	1.02 %	0.19 %	0.36 %	0.04 %	0.30 %	0.13 %
10 月	0.96 %	0.19 %	0.33 %	0.04 %	0.28 %	0.12 %
9 月	0.89 %	0.17 %	0.30 %	0.04 %	0.25 %	0.13 %
8 月	0.91 %	0.19 %	0.30 %	0.05 %	0.24 %	0.13 %
7 月	0.95 %	0.20 %	0.31 %	0.06 %	0.24 %	0.14 %
6 月	0.87 %	0.19 %	0.30 %	0.05 %	0.21 %	0.12 %
5 月	0.80 %	0.20 %	0.25 %	0.05 %	0.19 %	0.11 %
4 月	0.78 %	0.20 %	0.25 %	0.05 %	0.18 %	0.10 %
3 月	0.70 %	0.18 %	0.21 %	0.06 %	0.16 %	0.09 %
2 月	0.66 %	0.18 %	0.19 %	0.06 %	0.14 %	0.09 %
1 月	0.65 %	0.18 %	0.20 %	0.06 %	0.13 %	0.08 %

- iOS 主要指的是苹果移动设备（如 iPhone、iPad 和 iPod）的操作系统。

其他的统计数据

浏览器统计

屏幕分辨率统计

Internet Explorer 浏览器

Internet Explorer 浏览器



Internet Explorer 浏览器，简称 IE 浏览器，是微软公司（Microsoft）发布的一款免费的 web 浏览器。

Internet Explorer 发布于 1995 年，是当今最流行的浏览器之一。

下载 Internet Explorer

Internet Explorer 统计

下表是 浏览器统计信息 中关于 Internet Explorer 使用情况的细节：

2014	总计	IE 11	IE 10	IE 9	IE 8	IE 7	IE 6
5 月	8.9 %	2.7 %	1.4 %	1.9 %	2.1 %	0.2 %	0.0 %
4 月	9.4 %	2.7 %	1.6 %	2.0 %	2.4 %	0.3 %	0.1 %
3 月	9.7 %	2.6 %	1.6 %	2.1 %	2.5 %	0.3 %	0.1 %
2 月	9.8 %	2.5 %	1.7 %	2.1 %	2.7 %	0.3 %	0.1 %
1 月	10.2 %	2.5 %	1.7 %	2.3 %	3.1 %	0.4 %	0.1 %

2013	总计	IE 11	IE 10	IE 9	IE 8	IE 7	IE 6
12 月	9.0 %	0.8 %	2.1 %	2.4 %	3.2 %	0.4 %	0.1 %
11 月	10.5 %	3.5 %	2.5 %	3.6 %	0.5 %	0.1 %	
10 月	11.7 %	4.0 %	3.0 %	4.1 %	0.5 %	0.1 %	
9 月	12.1 %	4.0 %	2.8 %	4.6 %	0.6 %	0.1 %	
8 月	11.8 %	3.6 %	2.8 %	4.7 %	0.6 %	0.1 %	
7 月	11.8 %	3.3 %	2.8 %	4.8 %	0.7 %	0.1 %	
6 月	12.0 %	3.1 %	3.2 %	4.9 %	0.7 %	0.1 %	
5 月	12.6 %	2.6 %	3.9 %	5.2 %	0.8 %	0.1 %	
4 月	12.7 %	2.0 %	4.4 %	5.3 %	0.8 %	0.1 %	
3 月	12.9 %	1.3 %	5.2 %	5.5 %	0.8 %	0.2 %	
2 月	13.5 %	0.9 %	5.7 %	5.8 %	0.9 %	0.2 %	
1 月	14.3 %	0.8 %	5.9 %	6.4 %	1.0 %	0.3 %	

2012	总计	IE 10	IE 9	IE 8	IE 7	IE 6
12 月	14.7 %	0.6 %	5.9 %	6.8 %	1.1 %	0.3 %
11 月	15.1 %	0.4 %	6.5 %	6.8 %	1.1 %	0.3 %
10 月	16.1 %	0.2 %	6.8 %	7.6 %	1.2 %	0.3 %
9 月	16.4 %	0.2 %	6.6 %	7.9 %	1.3 %	0.4 %
8 月	16.2 %	0.1 %	6.1 %	7.8 %	1.8 %	0.5 %
7 月	16.3 %	0.1 %	5.9 %	7.9 %	2.0 %	0.6 %
6 月	16.7 %	0.1 %	6.1 %	8.0 %	1.9 %	0.6 %
5 月	18.1 %	0.1 %	6.5 %	8.8 %	2.1 %	0.6 %
4 月	18.3 %	0.1 %	6.4 %	8.8 %	2.3 %	0.7 %
3 月	18.9 %	0.0 %	6.1 %	9.4 %	2.5 %	0.9 %
2 月	19.5 %	0.0 %	5.7 %	10.2 %	2.6 %	1.0 %
1 月	20.1 %	0.1 %	5.3 %	10.5 %	3.1 %	1.1 %

2011	总计	IE 9	IE 8	IE 7	IE 6
12 月	20.2 %	5.1 %	10.7 %	3.2 %	1.2 %
11 月	21.2 %	5.1 %	11.5 %	3.4 %	1.2 %
10 月	21.7 %	5.1 %	11.8 %	3.5 %	1.3 %
9 月	22.9 %	4.8 %	12.4 %	3.9 %	1.8 %
8 月	22.5 %	4.2 %	11.9 %	4.2 %	2.0 %
7 月	22.0 %	3.9 %	11.7 %	4.1 %	2.3 %
6 月	23.2 %	3.6 %	12.9 %	4.4 %	2.3 %
5 月	24.9 %	3.1 %	14.1 %	5.3 %	2.4 %
4 月	24.3 %	2.1 %	14.8 %	4.9 %	2.5 %
3 月	25.8 %	1.1 %	16.3 %	5.4 %	3.0 %
2 月	26.5 %	0.6 %	16.7 %	5.7 %	3.5 %
1 月	26.6 %	0.5 %	16.6 %	5.7 %	3.8 %

2010	总计	IE 9	IE 8	IE 7	IE 6
12 月	27.5 %	0.5 %	16.5 %	6.1 %	4.4 %
11 月	28.6 %	0.4 %	17.6 %	6.5 %	4.1 %
10 月	29.7 %	0.4 %	17.3 %	7.2 %	4.8 %
9 月	31.1 %	0.2 %	17.3 %	8.0 %	5.6 %
8 月	30.7 %	16.2 %	7.8 %	6.7 %	
7 月	30.4 %	15.6 %	7.6 %	7.2 %	
6 月	31.0 %	15.7 %	8.1 %	7.2 %	
5 月	32.2 %	16.0 %	9.1 %	7.1 %	
4 月	33.4 %	16.2 %	9.3 %	7.9 %	
3 月	34.9 %	15.3 %	10.7 %	8.9 %	
2 月	35.3 %	14.7 %	11.0 %	9.6 %	
1 月	36.2 %	14.3 %	11.7 %	10.2 %	

2009	总计	IE 8	IE 7	IE 6
12 月	37.2 %	13.5 %	12.8 %	10.9 %
11 月	37.7 %	13.3 %	13.3 %	11.1 %
10 月	37.5 %	12.8 %	14.1 %	10.6 %
9 月	39.6 %	12.2 %	15.3 %	12.1 %
8 月	39.3 %	10.6 %	15.1 %	13.6 %
7 月	39.4 %	9.1 %	15.9 %	14.4 %
6 月	40.7 %	7.1 %	18.7 %	14.9 %
5 月	41.0 %	5.2 %	21.3 %	14.5 %
4 月	42.1 %	3.5 %	23.2 %	15.4 %
3 月	43.3 %	1.4 %	24.9 %	17.0 %
2 月	43.6 %	0.8 %	25.4 %	17.4 %
1 月	44.8 %	0.6 %	25.7 %	18.5 %

2008	总计	IE 7	IE 6	IE 5
12 月	46.0 %	26.1 %	19.6 %	0.3 %
11 月	47.0 %	26.6 %	20.0 %	0.4 %
10 月	47.4 %	26.9 %	20.2 %	0.3 %
9 月	49.0 %	26.3 %	22.3 %	0.4 %
8 月	51.0 %	26.0 %	24.5 %	0.5 %
7 月	52.4 %	26.4 %	25.3 %	0.7 %
6 月	54.2 %	27.0 %	26.5 %	0.7 %
5 月	54.4 %	26.5 %	27.3 %	0.6 %
4 月	54.8 %	24.9 %	28.9 %	1.0 %
3 月	53.9 %	23.3 %	29.5 %	1.1 %
2 月	54.7 %	22.7 %	30.7 %	1.3 %
1 月	54.7 %	21.2 %	32.0 %	1.5 %

2007	总计	IE 7	IE 6	IE 5
11 月	56.0 %	20.8 %	33.6 %	1.6 %
9 月	57.2 %	20.8 %	34.9 %	1.5 %
7 月	58.5 %	20.1 %	36.9 %	1.5 %
5 月	58.9 %	19.2 %	38.1 %	1.6 %
3 月	58.7 %	18.0 %	38.7 %	2.0 %
1 月	58.6 %	13.3 %	42.3 %	3.0 %

2006	总计	IE 7	IE 6	IE 5
11 月	60.6 %	7.1 %	49.9 %	3.6 %
9 月	62.1 %	2.5 %	55.6 %	4.0 %
7 月	62.4 %	1.9 %	56.3 %	4.2 %
5 月	63.0 %	1.1 %	57.4 %	4.5 %
3 月	64.7 %	0.6 %	58.8 %	5.3 %
1 月	66.0 %	0.2 %	60.3 %	5.5 %

2005	总计	IE 6	IE 5
11 月	68.9 %	62.7 %	6.2 %
9 月	75.5 %	69.8 %	5.7 %
7 月	73.8 %	67.9 %	5.9 %
5 月	71.6 %	64.8 %	6.8 %
3 月	72.5 %	63.6 %	8.9 %
1 月	74.5 %	64.8 %	9.7 %

2004	总计	IE 6	IE 5
11 月	76.2 %	66.0 %	10.2 %
9 月	79.0 %	67.8 %	11.2 %
7 月	80.4 %	67.2 %	13.2 %
5 月	81.9 %	68.1 %	13.8 %
3 月	82.8 %	68.2 %	14.6 %
1 月	84.7 %	68.9 %	15.8 %

2003	总计	IE 6	IE 5
11 月	84.9 %	71.2 %	13.7 %
9 月	86.6 %	69.7 %	16.9 %
7 月	87.2 %	66.9 %	20.3 %
5 月	87.7 %	65.0 %	22.7 %
3 月	88.0 %	63.4 %	24.6 %
1 月	84.6 %	55.3 %	29.3 %

2002	总计	IE 6	IE 5	IE4
11 月	83.4 %	53.5 %	29.9 %	
9 月	83.5 %	49.1 %	34.4 %	
7 月	85.0 %	44.4 %	40.1 %	0.5%
5 月	87.4 %	40.7 %	46.0 %	0.7%
3 月	86.8 %	36.7 %	49.4 %	0.7%
1 月	86.8 %	30.1 %	55.7 %	1.0%

以上统计数据是基于 W3CSchool 网站的用户。

Internet Explorer 11

针对 Windows 8.1：Internet Explorer 11 发布于 2013 年 10 月 17 日。

针对 Windows 7：Internet Explorer 11 发布于 2013 年 11 月 7 日。

新特性：

- 性能改进
- 全新的 **F12 Developer Tools**（**F12** 开发工具）
- **WebGL** 支持
- 高分辨率支持
- **CSS3 border-image** 支持
- **CSS Flexbox** 支持
- 支持 **SPDY**（仅限于 **Windows 8.1**）
- 支持全屏幕定位 **API**（**Full Screen and Orientation API**）
- 增强对基于触摸设备用户的支持
- **JavaScript** 增强
- **DOM** 突变观察
- 网络加密 **API**（**Web Cryptography API**）

- 视频文字跟踪支持
- 加密的媒体支持
- 改进的 **HTML** 编辑器

IE 11 删除的功能：

- **document.all** 被废弃 - 检查它是否存在的代码将无法检测到它，但是实际使用它的代码会继续工作。
- **attachEvent**
- 快速标签（**CTRL+Q**）
- **"Work offline"** （离线工作）命令从 **File**（文件）菜单移除
- 从 **IE** 拖放选中的内容到 **Word** 等其他程序
- 使用大图标命令按钮
- 在 **Developer Tools** 中一次性查看所有 **cookies** 的功能

IE 11 是快速和标准的兼容。然而，其主要缺点是，它不能运行在 Windows 7 之前的操作系统版本上，且不为该系统提供同步！

Internet Explorer 10

Internet Explorer 10 发布于 2012 年 9 月 4 日，是 Windows 8 中的默认浏览器。

在 2013 年 2 月，Internet Explorer 10 对所有 Windows 7 SP1 开放下载。

新特性：

- 比 **IE9** 快 **20%**
- 更新支持 **HTML5** - 现在支持 <script> 标签中的 async 属性、AppCache API、信道消息、拖放 API、历史、解析、沙盒（Sandbox）、拼写检查、视频、Web Workers 和 WebSockets。
- 更新支持 **CSS3** - 现在支持转换、动画、字体、渐变、过渡效果、文字阴影和样式表去除限制。
- 更新支持 **DOM** - 现在支持先进的命中测试 API、媒体查询监听、XMLHttpRequest 增强和指针手势事件。
- 更新支持 **SVG** - 现在支持 SVG 和过滤器。
- 更新支持索引数据库 **API**
- 集成的拼写检查和自动更正功能
- **Adobe Flash** 整合 - 包含一个内置的 Adobe Flash Player。
- 为触摸屏用户添加功能 - 全屏触摸浏览器。
- 向前翻转 - 通过在触摸屏设备上的滑动手势，快速移动到网站的下一个页面。
- 新的用户代理字符串 - Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; [platform token] Trident/6.0; Touch)

IE 10 不再 支持下列：

- **XML** 数据岛
- **VML** - 矢量标记语言（Vector Markup Language）。
- **HTC** - 元素的行为和 HTML 组件（Element behaviors and HTML components）。
- **DX** 过滤器 - 基于 DirectX 的过滤器/过渡效果。
- 条件注释 - 在 HTML 源代码中被 IE 解释的条件语句，如：<!--[if IE 6]>。用于向 IE 提供和隐藏代码。
- 内容顾问（**Content advisor**） - 由 Windows Parental Controls 接管。

Internet Explorer 9

Internet Explorer 9 发布于 2011 年 3 月。

新特性：

- 更新的界面 - 让您可以控制和聚焦网站查看。
- 轻快的速度 - IE9 比之前的 IE 版本快了很多。硬件加速的文本、视频和图形提高了网站运行的性能，就像程序已经安装在您的电脑上一样。同时，IE9 利用视频卡让一切变得更快。
- 把网站锁定到 **Windows 7** 的任务栏 - 微软与 Facebook、Twitter 和 Pandora 等众多网站有合作关系，您可以看到支持跳转列表的任务栏。
- 统一的地址栏 - 统一的地址栏，用于导航网站或开始一个搜索。
- 重新设计的新标签页 - 让您可以快速进入您经常访问的站点。您也可以重新打开关闭的标签，或者您的上次浏览会话，或开始一个新的非私人的浏览会话。
- 改进的附加组件管理 - 一个叫做附加组件性能顾问（Add-on Performance Adviser）的新的功能，让您了解是什么导致了您的浏览器速度变慢，并允许您删除它
- 更好的隐私保护功能 - 跟踪保护功能可以限制网站跟中您的信息。
- 更好地支持 **HTML5/CSS3** - IE9 也支持 SVG 和 画布图形（Canvas graphics）。
- 新的下载管理器 - 允许您查看通过浏览器下载的文件，如果文件是恶意的或包含病毒会进行通知。它还可以让您暂停下载，并在稍后恢复下载。
- 增**强**标签管理 - 您可以通过把标签从一个窗口拖动到另一个窗口，实现不同窗口间标签的移动。当您把标签放置在新的窗口后，它不需要重新加载页面。

Internet Explorer 8

Internet Explorer 8 发布于 2009 年 3 月。

新特性：

- 加速器 - 提供对网络服务的快速访问。用户能够从 web 查找并发送信息。举例来说，在一个餐馆网站，用户可以通过在网页内点击，就可以获得关于该餐厅的地图、新闻和信

息，有关它的 blog，或者在新浪微博上共享它。
- 网站订阅 - 是类似 RSS 的特性，允许用户直接在网页内订阅内容。
- 兼容性视图 - 在 IE7 中显示网页。
- 搜索建议 - 根据您的输入提供建议的内容。
- 新标签的功能 - 重新打开意外被关闭的标签，通过颜色代码分组相关的标签。
- 标签分离 - 防止因为一个错误的网站导致整个浏览器关闭，只有错误的标签页会被关闭。
- 非私人浏览 - 浏览网页时不保存任何数据（比如密码、cookies、浏览历史，等等）。
- 智能屏幕过滤器 - 保护您避免安装上恶意软件，恶意软件会暴露您的数据、隐私和身份，并有可能破坏您的计算机和有价值的数据。

IE 8 完全支持 CSS 2.1。此外，它修正了"许多跨浏览器的不一致性，比如 get/set/removeAttribute, default attributes, Attribute object 以及 <q> 标签"。

Internet Explorer 7

Internet Explorer 7.0 发布于 2006 年 11 月。IE7 是 Windows XP with Service Pack 2 中的标准浏览器。

Internet Explorer 7 提供改进的导航、通过工具栏进行 web 搜索、高级打印、即时搜索以及 RSS 提要。

新特性：

- 高级打印 - Internet Explorer 7 将自动缩放要打印的网页，以便整个网页都在打印的页面中。 打印选项还包括可调整的边距、可自定义的页面布局、可删除的页眉和页脚以及仅打印选定文本的选项。
- 即时搜索框 - 使用您喜爱的搜索提供程序的 Web 搜索现在可被添加到工具栏的搜索框中，消除了多个独立工具栏同时存在的混乱现象。 您可以方便地从下拉列表中选择提供程序或添加更多提供程序。
- 收藏中心 - 快捷方便地访问收藏夹、选项卡组、浏览历史记录和 RSS 提要订阅。 收藏中心可根据需要进行扩展，并可随意定位以方便访问。
- **RSS** 提要 - Internet Explorer 7 可自动检测站点上的 RSS 提要并使工具栏上的图标亮起。 您可以单击图标来任意预览和订阅 RSS 提要，那么您在内容更新后将自动获得通知。 您可以在浏览器中直接阅读 RSS 提要，浏览重要事件，以及使用搜索项或特定站点类别筛选视图。
- 选项卡式浏览 - 在单个浏览器窗口中查看多个站点。 通过浏览器框顶部的选项卡轻松地从一个站点切换到其他站点。
- 快速选项卡 - 通过在单个窗口中显示所有缩略图，您可以在打开的选项卡之间轻松选择和定位。
- 选项卡组 - 选项卡可按逻辑类别进行分组和保存，以便您单击一下鼠标即可打开多个选项卡。 选项卡组可方便地设为主页组，以便每次启动 Internet Explorer 时都可以打开整个选项卡组。

- 页面缩放 - 放大单个页面，包括文本和图片，以便您可以关注特定内容或者使视觉障碍者可以将内容看得更加清楚。

Internet Explorer 6

Internet Explorer 6.0 是 Windows XP 中的标准浏览器。发布于 2001 年 8 月。

Windows XP 基于 Windows 2000，是 Windows 98、Millennium 以及 Windows 2000 的继任者。

Internet Explorer 5

Internet Explorer 5 是首个支持 XML 的主流浏览器。

Version 5.5（发布于 2000 年 7 月），用于 Windows 95、98、NT 4.0 以及 Windows 2000。支持 XML/XSL、CSS、打印(打印预览)和 HTC 行为。

Version 5.01（发布于 1999 年 11 月），主要是修正了 5.0 的一个 bug。

Version 5.0（发布于 1999 年 3 月），是首个支持 XML 的主流浏览器。

更老的 Internet Explorer 版本

大多数用户都不在使用更老的 Internet Explorer 版本。Web 开发者会忽略它们。而它们的功能也过时了。

IE 4.0（发布于 1997 年）。这个版本对 CSS 和 DOM 的支持非常好，但不支持 XML。

IE 3.0（发布于 1996 年）的使用率不到 0.1%。

IE 2.0（发布于 1995 年）。这个版本太老了，已经没有人使用它了。

IE 1.0（发布于 1995 年）。还有人记得它吗？

Internet Explorer for Macintosh

Version 5.1.7 是最后一个用于 Mac OS 8 和 9 的 Internet Explorer，发布于 2003 年 7 月。

Version 5.2.3 是最后一个用于 Mac OS X 的 Internet Explorer，发布于 2003 年 6 月。

注释：version 5.2.3 需要 Mac OS X 中的特殊特性。Mac OS 8 和 9 的用户必须下载 version 5.1.7。

微软已在 **2005** 年 **12** 月 **31** 日停止对用于 **Mac** 的 **Internet Explorer** 的技术支持。

Mac 用户推荐使用其他浏览器，比如 苹果的 Safari 浏览器。

Microsoft Internet Explorer 资源

微软的 Internet Explorer 站点 微软的 Internet Explorer 支持站点 微软的支持站点包括了可搜索知识库、下载中心、产品 FAQ 以及辅助支持目录。

Mozilla Firefox 浏览器

Firefox 浏览器



Firefox 浏览器，中文名"火狐浏览器"，是来自 Mozilla 的一款免费的开源 web 浏览器。

Firefox 发布于 2004 年，也是当今最流行的浏览器之一。

Firefox 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

下载 Firefox

Firefox 统计

下表是 浏览器统计信息 中关于 Firefox 使用情况的细节：

2014	总计	FF 30	FF 29	FF 28	FF 27	FF 26	FF 25	更旧
5 月	24.9%	0.1%	14.2%	5.2%	0.5%	0.5%	0.2%	4.2%
4 月	25.0%	1.1%	18.1%	1.2%	0.7%	0.3%	3.6%	
3 月	25.6%	0.3%	3.4%	16.3%	1.0%	0.4%	4.2%	
2 月	26.4%	0.6%	11.7%	8.7%	0.6%	4.8%		
1 月	26.9%	1.2%	18.9%	1.0%	5.8%			

2013	总计	FF 26	FF 25	FF 24	FF 23	FF 22	FF 21	FF 20	FF 19
12 月	26.8%	6.9%	13.3%	1.1%	0.5%	0.4%	0.3%	0.4%	0.2%
11 月	26.8%	1.0%	16.2%	3.5%	0.7%	0.5%	0.4%	0.4%	0.3%
10 月	27.2%	0.1%	1.3%	18.4%	2.0%	0.7%	0.4%	0.5%	0.3%
9 月	27.8%	0.1%	0.3%	3.1%	17.5%	1.0%	0.6%	0.5%	0.3%
8 月	28.2%	0.7%	10.8%	9.9%	1.0%	0.7%	0.4%	0.4%	0.8%
7 月	28.9%	0.1%	1.2%	18.0%	3.1%	1.0%	0.5%	0.5%	0.8%
6 月	28.9%	0.1%	1.7%	19.8%	1.3%	0.7%	0.5%	0.8%	4.0%
5 月	27.7%	0.1%	0.5%	6.5%	14.0%	0.9%	0.6%	0.8%	4.3%
4 月	27.9%	0.1%	0.7%	12.9%	7.3%	0.8%	0.8%	5.9%	
3 月	28.5%	0.1%	0.9%	19.0%	1.4%	0.9%	6.2%		
2 月	29.6%	0.1%	0.2%	4.6%	16.0%	1.2%	7.5%		
1 月	30.2%	0.1%	0.6%	12.4%	8.3%	8.8%			

2012	总计	FF 17	FF 16	FF 15	FF 14	FF 13	FF 12	FF 11	FF 9-10
12 月	31.1%	16.3%	5.6%	1.3%	1.0%	0.7%	1.1%	0.6%	1.2%
11 月	31.2%	2.5%	19.4%	1.8%	1.2%	0.8%	1.2%	0.6%	1.3%
10 月	31.8%	0.4%	9.5%	12.4%	1.8%	1.0%	1.4%	0.8%	1.4%
9 月	32.2%	0.8%	17.7%	4.8%	1.3%	1.6%	1.0%	1.6%	1.9%
8 月	32.8%	1.9%	20.3%	2.1%	1.8%	1.1%	1.7%	2.1%	1.2%
7 月	33.7%	0.3%	7.4%	15.2%	3.0%	1.3%	2.1%	2.4%	1.5%
6 月	34.4%	0.6%	12.1%	12.5%	1.6%	2.3%	2.7%	1.8%	0.8%
5 月	35.2%	0.1%	0.8%	21.4%	3.8%	2.7%	3.4%	2.3%	0.7%
4 月	35.8%	0.1%	2.6%	21.6%	3.8%	3.8%	3.1%	0.8%	
3 月	36.3%	0.3%	9.1%	17.3%	5.0%	3.7%	0.9%		
2 月	37.1%	0.1%	0.6%	24.2%	6.5%	4.3%	1.4%		
1 月	37.2%	0.1%	18.6%	12.4%	5.0%	1.1%			

2011	总计	FF 9	FF 8	FF 7	FF 6	FF 5	FF 4	FF 3.6	FF 3.
12 月	37.7%	1.8%	22.8%	2.1%	1.5%	1.2%	1.3%	5.6%	0.6
11 月	38.1%	0.4%	12.7%	11.5%	2.2%	1.5%	1.5%	6.9%	0.6
10 月	38.7%	0.6%	19.8%	5.1%	2.0%	1.8%	7.9%	0.7%	0.5
9 月	39.7%	1.5%	22.2%	3.1%	2.2%	9.0%	0.7%	0.7%	0.3
8 月	40.6%	9.5%	15.9%	2.9%	10.3%	0.8%	0.8%	0.4%	
7 月	42.0%	0.6%	23.2%	4.6%	11.5%	0.9%	0.9%	0.3%	
6 月	42.2%	5.6%	21.5%	12.3%	1.5%	0.9%	0.4%		
5 月	42.4%	0.3%	23.4%	15.6%	1.8%	1.0%	0.3%		
4 月	42.9%	15.7%	23.8%	2.1%	1.1%	0.2%			
3 月	42.2%	5.2%	32.9%	2.5%	1.3%	0.3%			
2 月	42.4%	1.9%	35.8%	2.9%	1.5%	0.3%			
1 月	42.8%	1.5%	36.1%	3.1%	1.7%	0.4%			

2010	总计	FF 4	FF 3.6	FF 3.5	FF 3.0	更旧
12 月	43.5%	1.1%	36.5%	3.5%	2.0%	0.4%
11 月	44.0%	0.8%	36.9%	3.8%	2.0%	0.5%
10 月	44.4%	0.7%	36.2%	4.5%	2.4%	0.6%
9 月	45.1%	0.8%	35.3%	5.6%	2.9%	0.5%
8 月	45.8%	0.6%	35.2%	6.1%	3.1%	0.6%
7 月	46.4%	0.4%	34.5%	7.3%	3.6%	0.6%
6 月	46.6%	32.7%	9.1%	4.0%	0.8%	
5 月	46.9%	31.7%	10.0%	4.4%	0.8%	
4 月	46.4%	29.4%	11.5%	4.6%	0.9%	
3 月	46.2%	22.1%	17.6%	5.6%	0.9%	
2 月	46.5%	10.5%	28.5%	6.5%	1.0%	
1 月	46.3%	2.2%	34.2%	8.6%	1.3%	

2009	总计	FF 3.5	FF 3.0	FF 2.0	更旧
12 月	46.4%	33.3%	11.2%	1.2%	0.7%
11 月	47.0%	31.4%	13.6%	1.5%	0.5%
10 月	47.5%	29.8%	15.7%	1.6%	0.4%
9 月	46.6%	27.3%	17.2%	1.8%	0.3%
8 月	47.4%	21.0%	24.0%	1.9%	0.5%
7 月	47.9%	12.1%	33.3%	1.9%	0.6%
6 月	47.3%	1.2%	43.4%	2.3%	0.4%
5 月	47.7%	44.3%	2.9%	0.5%	
4 月	47.1%	43.4%	2.8%	0.9%	
3 月	46.5%	42.2%	3.0%	1.3%	
2 月	46.4%	41.5%	3.8%	1.1%	
1 月	45.5%	39.5%	4.6%	1.4%	

2008	总计	FF 3.0	FF 2.0	FF 1.5	Moz
12 月	44.7%	38.1%	6.2%	0.1%	0.3%
11 月	44.6%	35.9%	8.0%	0.3%	0.4%
10 月	44.5%	34.9%	8.7%	0.4%	0.5%
9 月	43.1%	31.9%	10.1%	0.6%	0.5%
8 月	44.2%	25.5%	17.5%	0.7%	0.5%
7 月	43.1%	20.7%	21.1%	0.8%	0.5%
6 月	41.5%	8.9%	31.2%	0.9%	0.5%
5 月	40.5%	2.9%	35.9%	1.0%	0.7%

以上统计数据是基于 W3CSchool 网站的用户。

Firefox 29

Firefox 29 发布于 2014 年 4 月 29 日。

Firefox 29 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 一些重新设计的工具和一个更新的界面
- 增强的火狐同步服务

- 菜单移到右上角
- 新的 "定制工具"，您可以在其中添加或者移动功能和附加组件
- 用户可以通过单击创建一个书签
- 支持 **CSS box-sizing** 属性（不带 **-moz-** 前缀）
- 支持 **HTML5 input type="number"**
- 支持 **HTML5 input type="color"**

Firefox 28

Firefox 28 发布于 2014 年 3 月 18 日。

Firefox 28 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- **Mac OS X**：通知中心（**Notification Center**）支持 **web** 通知
- 水平的 HTML5 音频/视频的音量控制
- 支持 **WebM** 中的 **Opus**
- 实现 **VP9** 视频解码
- 支持 **MathML 2.0 'mathvariant'** 属性
- 后台线程报告
- 支持布局中的多线 **flexbox**

Firefox 27

Firefox 27 发布于 2014 年 2 月 4 日。

Firefox 27 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 对 **Firefox Social API** 的一个重大更新 - 现在允许用户同时运行多个服务
- 支持 **Google** 的 **SPDY 3.1** 协议及传输层安全（**TLS - Transport Layer Security**）版本 **1.1** 和 **1.2** - 这些基本上是著名的 SSL 加密协议的接班者
- 使用 **"all:unset"** 重置样式表
- 支持滚动的字段集
- **CSS** 光标关键字 **-moz-grab** 和 **-moz-grabbing** 已没有前缀
- 支持数学函数 **Math.hypot()**
- 画布（**Canvas**）上支持虚线

Firefox 26

Firefox 26 发布于 2013 年 12 月 10 日。

Firefox 26 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 所有的 **Java** 插件默认为 **"**点击播放**"**
- 密码管理器支持脚本生成的密码域
- 在 **Linux** 上支持 **H.264** 编码的视频
- 在 **Windows XP** 上支持 **MP3** 音频 - 这完成了跨 Windows 操作系统版本的 MP3 支持
- 当站点请求 **AppCache** 时，移除提示
- 支持 **CSS3 image-orientation** 属性
- 新的应用程序管理器（**App Manager**） - 允许您在 Firefox 操作系统的手机和 Firefox 操作系统的模拟器上部署和调试 HTML5 程序
- **IndexedDB** 可用作一个 **"optimistic"** 的存储区 - 使用 LRU 驱逐策略，不要求在池中存储任何提示和数据，只是一个短暂的临时存储
- 减少内存使用，改进图像处理

Firefox 25

Firefox 25 发布于 2013 年 10 月 29 日。

Firefox 25 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 网络音频支持
- **CSS3 background-attachment:local** 支持
- **iframe** 文档内容可内联指定
- 许多新的 **ECMAScript 6** 函数都是可用的
- 探查器（**Profiler**）工具可以保存和导入测试结果

Firefox 24

Firefox 24 发布于 2013 年 9 月 17 日。

Firefox 24 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 启用对 **WebRTC** 的支持（在 **Android** 上）
- 在 **Android** 上共享 **NFC**
- 增加大规模关闭右侧标签功能
- 在 **OS X 10.7** 上新的滚动条样式

- 将聊天窗口拖拽成独立窗口
- 优化在图像平铺和缩放时的图像周围的显著 **SVG** 渲染
- 为增强的调试体验改进浏览器控制台，取代现有的错误控制台
- 不再支持撤销列表功能
- 不再支持从应用程序或配置文件目录加载的 **sherlock** 文件

Firefox 23

Firefox 23 发布于 2013 年 8 月 7 日。

Firefox 23 可以在 Windows、Mac、Linux 和 Android 手机版上运行。

新特性：

- 新的 **logo**
- 混合的内容阻止程序 **-** 通过完全限制 **URL**，来防止阅读网络上不安全的内容
- 分享按钮 **-** 让用户只需要通过一个简单的点击，就可以把内容分享给朋友**/**家人
- 专门为开发人员设计的网络监控工具 **-** 分解网站组件，并通知开发人员加载的时间

Firefox 22

Firefox 22 发布于 2013 年 6 月 25 日。

Firefox 22 可以在 Windows、Mac、Linux 和 Android 手机版上运行。

新特性：

- 修复 **14** 个安全问题
- 默认情况下，启用 **WebRTC**
- **OdinMonkey -** 改进 **JavaScript** 性能
- 可以改变 **HTML5** 音频**/**视频播放速率
- 新的显示缩放选项，默认情况下是启用的 **-** 在高分辨率显示器上呈现较大的文本
- 文本文件的自动换行

Firefox 21

Firefox 21 发布于 2013 年 5 月 14 日。

Firefox 21 可以在 Windows、Mac、Linux 和 Android 手机版上运行。

新特性：

- 火狐健康报告（**Firefox Health Report**） **-** 一个调整浏览器的工具

- 改进的启动时间
- 支持 **HTML5 <main>** 元素
- 支持带作用域的样式表
- 三个 **"不跟踪"** 的选项：**"跟踪"**、**"不跟踪"** 和 **"未设定"**
- 扩展的社交 **API**（**Social API**） **-** 增加新的社交服务：**Cliqz**、**Mixi** 和 **msnNOW**，以及 **Firefox** 的 **Facebook Messenger**（一种桌面窗口聊天客户端）

Firefox 20

Firefox 20 发布于 2013 年 4 月 2 日。

Firefox 20 可以在 Windows、Mac、Linux 和 Android 手机版上运行。

新特性：

- 新的下载管理器
- 私人浏览窗口
- 在单独的窗口中查看开发工具
- 在浏览器不挂起的情况下关闭挂起的插件
- **<canvas>** 现在支持混合模式
- 各种 **<audio>** 和 **<video>** 的改进

Firefox 19

Firefox 19 发布于 2013 年 2 月 20 日。

Firefox 19 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 新的内置的 **PDF** 阅读器 - 不需要插件即可阅读 PDF
- **@page** 支持打印的文档
- 导出画布内容为一个图像 - canvas 元素的内容可通过 toBlob() 导出为一个图像
- 实现 **CSS** 视口**-**百分比长度单位（**vh**、**vw**、**vmin** 和 **vmax**）
- 支持 **CSS text-transform:full-width**
- 更快的启动
- **XForms** 已被移除

Firefox 18

Firefox 18 发布于 2013 年 1 月 8 日。

Firefox 18 可以在 Windows、Mac OS X（Snow Leopard、Lion 和 Mountain Lion）、Linux 和 Android 上运行。

新特性：

- **IonMonkey** - 新的 JavaScript 编译器（比 Firefox 17 快了 7% - 26%）
- 支持视网膜显示
- 内置的 **PDF** 查看器
- 初步支持网络实时通信（**WebRTC - Web Real Time Communication**）

Firefox 17

Firefox 17 发布于 2012 年 11 月 20 日。

Firefox 17 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 更新 **Awesome Bar** - 带有更大的图标
- 标签动画
- 社交 **API**（**Social API**） **-** 允许您通过浏览器登录到您的社交网络（Facebook）
- 实现单击播放 **-** 为了防止弱势插件版本在未经过用户允许的情况下运行/li>
- 为页面检查器中的 **HTML/DOM** 增加新的标记面板
- 实现 **HTML5** 沙盒（**sandbox**）属性（用于 **iframes**）
- 支持 **Mountain Lion** 的通知中心（**Notification Center**）

Firefox 16

Firefox 16 发布于 2012 年 10 月 9 日。

Firefox 16 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 本地支持 **PDF**
- 新的开发工具栏 - 允许您访问 Web 控制台、检查器和调试器。工具栏本身支持一些命令
- 针对 **OS X** 的 **VoiceOver** 支持
- **Web** 应用程序的支持 - 开始使用在 Mozilla app 目录 中的应用程序
- 增量垃圾回收（**Incremental Garbage Collection**） - 加快您的浏览器，回收/重复使用 JavaScript 程序不再使用的内存
- 在 **Firefox 16** 中支持无前缀化的 **CSS3** 动画、过渡效果（**Transitions**）、转换（**Transforms**）和渐变（**Gradients**）。

Firefox 15

Firefox 15 发布于 2012 年 8 月 28 日。

Firefox 15 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 无缝背景更新
- 停止由附加组件引起的大多数内存泄漏
- 支持 **SPDY** 网络协议 **v3**
- **WebGL** 增强功能
- **HTML5 -** 支持 **<source>** 中的 **media** 属性
- **HTML5 -** 支持 **<audio>** 和 **<video>** 中的 **played** 属性
- **CSS3 -** 支持 **CSS word-break** 属性
- 更快的调试器
- 新的设计工具，允许 **Web** 开发人员在桌面和移动的站点视图之间进行切换
- 本地支持 **Opus** 音频编解码器

Firefox 14

Firefox 14 发布于 2012 年 7 月 17 日。

Firefox 14 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 隐私特征 - Firefox 使用 HTTPS 加密您的 Google 搜索
- **Bug** 修复 - 为 Mac OS X Lion 提供全屏支持
- 安全特性 - 您可以把经常访问的站点和信任的站点放入白名单

Firefox 13

Firefox 13 发布于 2012 年 6 月 5 日。

Firefox 13 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 默认的主页能更快地访问书签、历史、设置等
- 当打开一个新标签时，用户可以看到他们最常访问的网页
- 默认情况下，启用 **SPDY** 协议，以便更快地浏览被支持的站点
- 总计 **72** 个改进，包括页面检查器（**Page Inspector**）、**HTML** 面板、样式检查器（**Style Inspector**）、暂存器（**Scratchpad**）和样式编辑器（**Style Editor**）

- 改进支持下列的 **CSS** 属性：**column-fill**、**CSS3 background-position**
- 支持 **:invalid** 伪类
- 现在支持 **CSS3 <angle>** 类型单位

Firefox 12

Firefox 12 发布于 2012 年 4 月 24 日。

Firefox 12 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- **Windows**：**Firefox** 通过一个小提示更容易进行更新（用户帐户控制）
- 页面源代码显示行号
- 在 **title** 属性中支持换行
- 支持 **text-align-last CSS** 属性

Firefox 11

Firefox 11 发布于 2012 年 3 月 13 日。

Firefox 11 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 样式编辑器（**Style Editor**） **-** 一个新的样式表编辑器。访问任意网页，然后从 **Web Developer** 菜单选择样式编辑器（**Style Editor**）
- **"**倾斜的**" 3D** 页面结构视图 **-** 检查工具提供一个 **"3D"** 按钮
- 支持 **CSS text-size-adjust** 属性（控制移动设备文本大小，并在放大到一个网页时显示滚动条）
- 支持 **outerHTML** 属性
- **Firefox** 可以从 **Google Chrome** 迁移进书签、历史和 **cookies**
- 当查看源代码时，**HTML5** 标签可以正确地高亮显示
- 文件可以存储在 **IndexedDB** 中
- 移除 **Websockets API** 的 **moz** 前缀

Firefox 10

Firefox 10 发布于 2012 年 1 月 31 日。

Firefox 10 可以在 Windows、Mac OS X、Linux 和 Android 上运行。

新特性：

- 页面检查器（**Page Inspector**）**/CSS** 检查器（**CSS Inspector**） **-** 让开发人员检查 **HTML** 和 **CSS**
- 暂存器（**Scratchpad**） **-** 为 **JavaScript** 开发人员高亮显示语法的代码编辑器
- 新的 **3D** 图形处理能力和 **WebGL** 内容的反走样
- 全屏 **API -** 使开发人员能够创建全屏应用程序和游戏，传送全屏视频内容
- 支持 **CSS 3D** 转换

Firefox 9

Firefox 9 发布于 2011 年 12 月 20 日。

Firefox 9 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 增加了类型推理（**Type Inference**），改进了 **JavaScript** 性能（比 **Firefox 8** 快了 **30%**）
- 针对 **Mac OS X Lion** 改进的主题整合
- 为 **Mac OS X Lion** 增加了两个手指滑动导航
- 增加了通过 **JavaScript** 查询 **"**不跟踪**"** 状态的支持
- 增加了对 **CSS3 font-stretch** 的支持
- 改进了对 **HTML5**、**MathML** 和 **CSS** 的支持

Firefox 8

Firefox 8 发布于 2011 年 11 月 8 日。

Firefox 8 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 默认情况下安装 **Twitter** 搜索引擎
- 默认情况下禁用第三方插件
- 一个新的 **"**附加组件选择对话框**"**，允许在升级时验证插件
- 标签可 **"**在需求时**"** 被加载，让它更快地存储带有多个标签的窗口
- 支持跨域资源共享（**CORS**，**Cross-Origin Resource Sharing**），让开发者以一种安全的方式从其他域加载 **WebGL** 纹理
- **HTML5** 上下文菜单支持
- 用于 **Android** 的 **Firefox**：**Mozilla** 也添加了更多的功能，以便 **Firefox** 浏览器能运行在 **Android** 设备上

Firefox 7

Firefox 7 发布于 2011 年 9 月 27 日。

Firefox 7 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 改进的内存处理
- 当使用 **Firefox** 同步时，书签和密码更改总是即时同步。
- **"http://" URL** 前缀默认情况下是隐藏的（就像 **Chrome** 一样）
- 把 **WebSocket** 协议从版本 **7** 更新到版本 **8**
- 为用户添加一个系统来把性能数据发送回 **Mozilla**

Firefox 6

Firefox 6 发布于 2011 年 8 月 16 日。

Firefox 6 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 地址栏会高亮显示您所访问的网站的域
- 流线型的站点标识块外观
- 通过一个前缀的 **API**，增加了对 **WebSockets** 最新草案版本的支持。
- 增加了对 **EventSource / server-sent** 事件的支持
- 增加了对 **window.matchMedia** 的支持
- 增加了 **Scratchpad**，一个交互式的 **JavaScript** 原型环境
- 增加了一个新的 **Web Developer** 菜单项，移动相关开发项到其中
- 提高 **Web** 控制台（**Web Console**）的可用性
- 提高 **Firefox** 同步的发现能力
- 当使用全景图（**Panorama**）时，减少浏览器的启动时间。

Firefox 5

Firefox 5 发布于 2011 年 6 月。

Firefox 5 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 添加 **CSS** 动画支持
- 调谐 **HTTP** 空闲连接逻辑以提高性能
- 改进的画布（**canvas**）、**JavaScript**、内存和网络性能
- 改进支持 **HTML5**、**XHR**、**MathML**、**SMIL** 和画布（**canvas**）的标准。
- 改进的拼写检查

- 为 **Linux** 用户改进桌面环境的整合
- **WebGL** 内容不再加载跨域的纹理（**textures**）
- 背景标签限制了 **setTimeout** 和 **setInterval** 为 **1000ms**，以便提高性能

Firefox 4

Firefox 4 发布于 2011 年 3 月。

Firefox 4 可以在 Windows、Mac OS X 和 Linux 上运行。

新特性：

- 支持超过 **80** 种语言！
- 曾经最快的 **Firefox -** Firefox 比以前发布的版本的速度快了六倍。
- 应用程序标签 - 一个永久的主页，显示经常访问的站点，比如 Web 邮件、Twitter、Pandora、Flickr。
- 切换到标签 - 很容易地找到并切换到 Awesome Bar 上的任意打开的标签。
- 展开图（**Panorama**） - 拖放标签到可管理组以便在导航多个打开的标签时节省时间。
- **Firefox** 同步 - 跨多台计算机和移动设备访问您的 Awesome Bar 历史、书签、打开的标签、密码和数据。
- 新的附加组件管理器 - 超过 200,000 个附加组件，可用于定制 Firefox 的特性、功能和外观。
- 不跟踪 - 允许用户选择退出用于行为广告的跟踪。
- 隐私第一 - 防止其他人访问您的浏览器的历史。
- **HTTP** 严格的安全运输（**HSTS，HTTP Strict Transport Security**） - 建立安全连接来停止 "中间人" 的攻击，从而保护敏感数据的安全。
- 内容安全策略（**CSP，Content Security Policy**） - 通过允许网站明确告诉浏览器哪些内容是合法的，来防止交叉脚本的攻击。
- **JIT** 编译器 - 随着增强现有的 TraceMonkey JIT 和 SpiderMonkey 的解释器，获得更快的页面加载速度和更好的性能。
- **HTML5** 支持 - 包括硬件加速、高清视频（WebM）、3D 图形、离线数据存储、专业排版、触摸屏界面和 Mozilla Audio API。
- 改进了现有的工具 - 比如 CSS、画布（Canvas）和 SVG。
- 不间断的浏览 - 在 Adobe Flash、Apple QuickTime 或 Microsoft Silverlight 插件崩溃时，不间断用户的浏览。

Firefox 3.6

Firefox 3.6 发布于 2011 年 1 月。

Firefox 3.6 是建立在 Mozilla 的 Gecko 1.9.2 网页渲染平台上，该平台自 2009 年初开始开发，为 Web 开发人员、插件开发者及用户包含了许多改进功能，这个版本也比以前的版本更快，更具响应性，且已经进行优化，可运行在小型设备操作系统上，比如 Maemo。

新特性：

- 人物角色的主题 - 允许用户通过单击改变 Firefox 的外观
- 防止安装过时的插件 - 当用户浏览网页时，保证用户安全
- 打开本地视频可以全屏显示，支持公告框架
- 改进的 **JavaScript** 性能
- 为 **Web** 开发者提供异步运行脚本的能力 - 为了加快页面加载时间
- 支持可下载的 **Web** 字体 - 使用新的 WOFF 字体格式
- 改进对 **CSS3**、**DOM** 和 **HTML5** 的支持

更旧的 Firefox 版本

Firefox 3.5 - 发布于 2009 年 6 月。

Firefox 3.0 - 发布于 2008 年 6 月。

Firefox 2.0 - 发布于 2006 年 12 月。

Firefox 1.5 - 发布于 2005 年 11 月。

Firefox 1.0 - 发布于 2004 年 11 月。

Firefox 资源

Firefox 的支持站点 Firefox 的支持站点包括了可搜索知识库、下载中心、产品 FAQ 以及辅助支持目录。

Google Chrome 浏览器

Google Chrome 浏览器



Google Chrome 浏览器，中文名"谷歌浏览器"，是一款免费的开源 web 浏览器，它由 Google 开发，发布于 2008 年。

当 Google 决定开发一款浏览器时，他们需要彻底地重新谋划这款浏览器，这是因为如今的浏览器与仅需要浏览简单的文本页面时有很大的不同，现在，我们在浏览器上发邮件、购物、付账单，以及运行其他的大型应用程序。

Google Chrome 是当今最常用的浏览器：下载 Chrome。

Google Chrome 统计

下表是 浏览器统计信息 中关于 Google Chrome 使用情况的细节：

2014	总计	C 35	C 34	C 33	C 32	C 31	C 30	更旧
5 月	59.2%	12.8%	40.5%	1.8%	0.8%	0.8%	0.3%	2.2%
4 月	58.4%	31.7%	21.4%	1.0%	0.9%	0.3%	3.1%	
3 月	57.5%	0.8%	49.8%	2.2%	1.3%	0.4%	3.0%	
2 月	56.4%	0.4%	12.2%	33.3%	2.0%	0.5%	8.0%	
1 月	55.7%	26.4%	24.5%	0.8%	4.0%			

2013	总计	C 32	C 31	C 30	C 29	C 28	C 27	C 26	C 25
12 月	55.8%	1.0%	49.0%	1.6%	0.8%	0.8%	0.6%	0.3%	0.2%
11 月	54.8%	0.5%	26.3%	23.7%	1.1%	1.0%	0.6%	0.4%	0.3%
10 月	54.1%	0.2%	0.5%	39.8%	8.3%	1.9%	0.7%	0.4%	0.4%
9 月	53.2%	0.6%	46.0%	3.2%	1.0%	0.5%	0.4%	0.2%	0.2%
8 月	52.9%	0.5%	11.1%	36.5%	1.9%	0.5%	0.5%	0.3%	0.2%
7 月	52.8%	0.2%	0.5%	30.3%	18.3%	0.8%	0.6%	0.3%	0.3%
6 月	52.1%	0.4%	1.5%	45.3%	1.7%	1.1%	0.4%	0.3%	1.4%
5 月	52.9%	0.5%	10.0%	38.4%	1.4%	0.5%	0.4%	1.7%	
4 月	52.7%	0.3%	0.7%	45.8%	2.9%	0.7%	0.6%	1.7%	
3 月	51.7%	0.4%	2.1%	44.2%	2.1%	0.8%	2.1%		
2 月	50.0%	7.7%	37.8%	1.4%	3.1%				
1 月	48.4%	0.7%	26.5%	17.7%	3.5%				

2012	总计	C 25	C 24	C 23	C 22	C 21	C 20	C 19	C 18
12 月	46.9%	0.4%	0.7%	41.7%	1.1%	0.5%	0.5%	0.2%	0.5%
11 月	46.3%	0.3%	0.6%	30.3%	10.7%	1.1%	0.7%	0.4%	0.7%
10 月	44.9%	0.3%	0.7%	38.2%	1.8%	0.9%	0.5%	0.8%	0.4%
9 月	44.1%	0.4%	4.1%	35.2%	0.9%	0.6%	1.0%	0.4%	0.3%
8 月	43.7%	0.1%	0.5%	34.7%	4.2%	0.9%	1.2%	0.5%	0.3%
7 月	42.9%	0.3%	0.7%	35.4%	2.7%	1.3%	0.6%	0.4%	1.5%
6 月	41.7%	0.4%	1.6%	35.4%	1.5%	0.7%	0.4%	1.7%	
5 月	39.3%	0.1%	0.5%	15.8%	19.8%	0.9%	0.5%	1.7%	
4 月	38.3%	0.3%	0.9%	31.3%	3.4%	0.6%	1.8%		
3 月	37.3%	0.6%	1.5%	31.9%	1.0%	2.3%			
2 月	36.3%	0.3%	0.9%	17.1%	15.3%	2.7%			
1 月	35.3%	0.3%	1.1%	30.5%	3.4%				

2011	总计	C 17	C 16	C 15	C 14	C 13	C 12	C 11	C 10
12 月	34.6%	0.7%	13.4%	17.8%	0.9%	0.4%	0.4%	0.2%	0.2%
11 月	33.4%	0.4%	1.0%	28.1%	1.6%	0.5%	0.5%	0.2%	0.2%
10 月	32.3%	0.5%	3.3%	25.5%	0.9%	0.8%	0.3%	0.3%	0.1%
9 月	30.5%	0.7%	11.2%	16.0%	1.1%	0.3%	0.3%	0.1%	0.1%
8 月	30.3%	0.3%	1.0%	21.2%	6.0%	0.5%	0.3%	0.1%	0.2%
7 月	29.4%	0.7%	1.2%	25.3%	0.7%	0.4%	0.2%	0.2%	0.7%
6 月	27.9%	0.2%	0.8%	17.3%	7.6%	0.8%	0.2%	0.2%	0.8%
5 月	25.9%	0.3%	1.1%	21.0%	2.2%	0.3%	0.3%	0.7%	
4 月	25.6%	0.6%	1.9%	21.3%	0.6%	0.4%	0.8%		
3 月	25.0%	0.1%	0.8%	14.8%	7.7%	0.6%	1.0%		
2 月	24.1%	0.2%	0.9%	17.0%	4.8%	1.2%			
1 月	23.8%	0.7%	1.3%	20.4%	1.4%				

2010	总计	C 9	C 8	C 7	C 6	C 5	C 4	更旧
12 月	22.4%	1.2%	16.6%	3.5%	0.4%	0.4%	0.2%	0.1%
11 月	20.5%	0.6%	0.9%	17.7%	0.7%	0.4%	0.2%	0.0%
10 月	19.2%	0.3%	5.6%	12.3%	0.6%	0.2%	0.2%	
9 月	17.3%	0.6%	11.4%	4.9%	0.3%	0.1%		
8 月	17.0%	0.1%	1.3%	15.1%	0.3%	0.2%		
7 月	16.7%	0.8%	15.3%	0.4%	0.2%			
6 月	15.9%	0.7%	13.9%	1.1%	0.2%			
5 月	14.5%	0.3%	3.4%	10.6%	0.2%			
4 月	13.6%	2.5%	10.8%	0.3%				
3 月	12.3%	1.7%	10.4%	0.2%				
2 月	11.6%	1.0%	10.0%	0.6%				
1 月	10.8%	2.8%	8.0%					

2009	总计	C 4	C 3	C 2	C 1
12 月	9.8%	1.9%	7.7%	0.1%	0.1%
11 月	8.5%	1.1%	7.1%	0.2%	0.1%
10 月	8.0%	0.5%	7.0%	0.4%	0.1%
9 月	7.1%	0.4%	2.7%	3.9%	0.1%
8 月	7.0%	0.8%	6.1%	0.1%	
7 月	6.5%	0.3%	6.0%	0.2%	
6 月	6.0%	0.2%	4.6%	1.2%	
5 月	5.5%	0.8%	4.6%		
4 月	4.9%	0.4%	4.4%		
3 月	4.2%	0.2%	3.9%		
2 月	4.0%	0.1%	3.8%		
1 月	3.9%	0.1%	3.7%		

2008	总计	C 1	Beta
12 月	3.6%	1.6%	2.0%
11 月	3.1%	3.1%	
10 月	3.0%	3.0%	
9 月	3.1%	3.1%	

以上统计数据是基于 W3CSchool 网站的用户。

Google Chrome Comic（漫画）

Google 制作了一个关于他们如何开发 Google Chrome 浏览器的漫画：

http://www.google.com/googlebooks/chrome/

Google Chrome 34

Chrome 34 发布于 2014 年 4 月 8 日。

Chrome 34 可以在 Windows、Mac、Linux 和 Android 上运行（4 月 2 日）。

新特性：

- 在设置了 **autocomplete=off** 时，**Chrome** 也会记住并填充密码域。

- 支持响应式的图像 **-** Google 推出了 "srcset" 属性，让网站开发者可以为一个图像提供不同分辨率的多个资源。总之，这意味着浏览器会选择与设备相匹配的资源，这些设备可以是台式机、笔记本电脑、平板电脑、手机、电视。
- **Web Audio API** 的无前缀的版本
- 把受监督的用户导入到新电脑 **-** 导入受监督用户时会同时导入他们的所有权限，这将会自动同步到设备。
- 许多新的应用程序**/**扩展的 **API**
- 一个针对 **Win8 Metro** 模式的不同的外观
- **31** 个安全修复

Google Chrome 33

Chrome 33 发布于 2014 年 2 月 20 日。

Chrome 33 可以在 Windows、Mac、Linux 和 Android 上运行（2 月 26 日）。

Chrome Frame（针对 Internet Explorer 的 Chrome 窗口插件）已正式退休。

新特性：

- **Web Speech API -** 语音识别与合成功能
- 自定义元素 **-** 您可以创建新的 HTML 和 DOM 元素（您可以使用任意您想使用的名称），还可以从其他元素扩展新的元素，并且捆绑自定义功能到一个单一的元素。
- 在离线模式下，可以创建、重命名、删除谷歌驱动（**Google Drive**）中的文件夹
- 在离线模式下，可以重命名谷歌驱动（**Google Drive**）中的文件
- **28** 个安全修复

Google Chrome 32

Chrome 32 发布于 2014 年 1 月 14 日。

Chrome 32 可以在 Windows、Mac、Linux、Chrome Frame 和 Android 上运行（1 月 15 日）。

新特性：

- 标签指标（声音 **sound**、摄像头 **webcam**、投射 **casting**）
- 一个针对 **Win8 Metro** 模式的新的外观
- 自动阻塞恶意软件的下载
- 监督用户 **- Chrome** 用户现在可以为家庭成员创建子账户，以便限制和监控网站访问（访问 **chrome.com/manage**）。
- **11** 个安全修复

Google Chrome 31

Chrome 31 发布于 2013 年 11 月 12 日。

Chrome 31 可以在 Windows、Mac、Linux、Chrome Frame 和 Android 上运行（11 月 15 日）。

新特性：

- 支持网络支付 **-** 一个新的 **API**，允许用户存储支持常用的付款细节，如信用卡号码
- 推出便携式本地客户端（**PNaCl**，**Launches Portable Native Client**）**-** 这个工具可以让开发者一旦在任何硬件和网站运行代码时，即可编译他们的代码。
- **25** 个安全修复

Google Chrome 30

Chrome 30 发布于 2013 年 10 月 1 日。

Chrome 30 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 搜索图像 **-** 在图像上右击并选择 **"**谷歌搜索这个图像**"**
- 三个新的触摸屏手势，包括水平滑动来切换标签，以及拖拽工具栏中的按钮可以看到标签切换视图
- 新的 **API**，比如 **MediaSource** 和 **DeviceMotion**（获得更好的应用程序和设备之间的集成）
- **50** 个安全修复

Google Chrome 29

Chrome 29 发布于 2013 年 8 月 20 日。

Chrome 29 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 改进的 **Omnibox** 建议（基于最近访问过的网站）
- 您可以重置您的权限回初始状态
- 更多新的应用程序和扩展 **API**
- 稳定性和性能方面的改进
- **25** 个安全修复

Google Chrome 28

Chrome 28 发布于 2013 年 7 月 30 日。

Chrome 28 可以在 Windows、Mac、Linux 和 Android 上运行。

新特性：

- 使用 **Blink** 内核作为渲染引擎（不再使用 **Webkit**）
- 支持 **CSS @support** 规则
- 丰富了对 **Chrome** 应用程序和扩展的通知
- **Android** 上的 **HTML5** 全屏 **API**
- **16** 个安全修复

Google Chrome 27

Chrome 27 发布于 2013 年 5 月 22 日。

Chrome 27 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 比之前版本快了 **5%** 的页面加载
- 支持下列 **HTML5** 输入类型：**date**、**datetime-local**、**month**、**week** 和 **time**
- 开发者工具（**Developer Tools**）窗口可向右停靠
- **Sync FileSystem API**（同步文件系统 **API**）**-** 一个新的离线存储 **API**
- **14** 个安全修复

Google Chrome 26

Chrome 26 发布于 2013 年 3 月 26 日。

Chrome 26 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 改进的拼写检查
- **Windows** 上多用户（权限）的桌面快捷方式
- 为 **Chrome** 中所有语言更新词典
- 支持三种新的语言（韩语 **Korean**、泰米尔语 **Tamil**、阿尔巴尼亚语 **Albanian**）

Google Chrome 25

Chrome 25 发布于 2013 年 2 月 21 日。

Chrome 25 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 语音识别支持 - 通过 Web Speech API
- 更好地支持 **HTML5 time/date** 输入
- 包括 **Adobe Flash** 更新
- 在扩展桌面模式中支持多监视器
- 为 **Windows** 用户移除默认的扩展安装

Google Chrome 24

Chrome 24 发布于 2013 年 1 月 10 日。

Chrome 24 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 速度的改进
- **HTML5 <datalist>** 元素支持日期和时间
- 增加对 **MathML** 的支持
- 包括了对 **CSS Custom Filters** 的实验支持
- **Bug/**安全修复

Google Chrome 23

Chrome 23 发布于 2012 年 11 月 6 日。

Chrome 23 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 提供了一个 **"不跟踪"** 的特性
- 包括了 **Windows** 上的 **GPU** 加速的视频解码 **-** 当观看视频时明显增加了电池的寿命
- 包括了更简单的网站权限

Google Chrome 22

Chrome 22 发布于 2012 年 9 月 25 日。

Chrome 22 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- **Pointer Lock API (Mouse Lock) -** 使用鼠标时更精确的定位
- **Windows 8** 增强功能
- 苹果的视网膜显示屏的改进
- 移除旧的集成的 **Flash** 插件（**gcswf32.dll**） **-** 现在只有完整的 **Flash Player** 插件 **PepperFlash**，这是一个跨平台的 **web** 浏览器插件 **API**。
- **24** 个安全修复

Google Chrome 21

Chrome 21 发布于 2012 年 8 月 1 日。

Chrome 21 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 支持苹果的视网膜显示
- 支持 **getUserMedia JavaScript API**（允许 **Web** 应用程序访问用户的摄像头和麦克风）
- 改进的对手柄输入设备的支持
- 与谷歌的云打印服务更深层次的整合
- **26** 个安全修复

Google Chrome 20

Chrome 20 发布于 2012 年 7 月 11 日。

Chrome 20 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 各种的修正和稳定性方面的改进
- **"**新的标签**"** 按钮更大了

Google Chrome 19

Chrome 19 发布于 2012 年 5 月 16 日。

Chrome 19 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 标签同步 - 当您登录到 Chrome，您的打开标签会在您所有的设备中同步。
- **Chrome** 启动标签 - 默认情况下，Chrome 设置启动标签（当您打开 Chrome 时看到的第

一个页面）为 "新的标签" 页。

Google Chrome 18

Chrome 18 发布于 2012 年 4 月 19 日。

Chrome 18 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- **HTML5** 中改进的 **2D** 图形功能
- 包括软件光栅 - 为了使更旧的机器显示更新的机器上的内容，启用不支持 WebGL 等技术。

Google Chrome 17

Chrome 17 发布于 2012 年 2 月 8 日。

Chrome 17 可以在 Windows、Mac 和 Linux 上运行。

新特性：

- 下载扫描保护 - 通过比对 Google 列表，检查下载的 .exes 和 .msi 文件是否是从一个已知的恶意网站下载的。
- **Omnibox** 预渲染网页 - 浏览器会在您完成输入之前，提前在后台加载搜索结果/网页，这让网页加载看起来更快了。

注意：2012 年 2 月 6 日 - Google 为 Android 4.0 平板电脑和手机发布了第一个 Chrome 测试版本！

Google Chrome 16

Chrome 16 发布于 2011 年 12 月 14 日。

新特性：

- 新的同步工具 - 登录到 Chrome，让您可以把您的东西保存到账号下。您可以把书签、扩展、应用程序和设置等保存到您的谷歌帐户，这样您就能在任何计算机上使用您所保存的东西。这也是一种很好的在线备份的方式，即使您的电脑死机了，您也能够轻松地恢复（同步）浏览器数据和设置到一台新的计算机。
- 在一个浏览器窗口中多个配置的用户支持 - 这使得多个同步的用户可以使用相同的浏览器窗口，加载他们的浏览历史/书签。
- **Google Cloud Print**（谷歌云打印） - 使用谷歌云打印可以打印任何网页（您的打印机

允许您访问任何启用的 web 应用程序）。

Google Chrome 15

Chrome 15 发布于 2011 年 10 月 25 日。

新特性：

- 重新设计 **"**新的标签**"** 页 - 让您可以在应用程序和频繁访问的网站之间切换。
- 扩展管理器 - 现在是选项页面的一部分，且有一个新的界面。
- 默认情况下启用 **JavaScript** 全屏 **API** - 支持全屏的 HTML5 视频。
- 在菜单中增加 **Chrome** 同步
- 更快的打印预览

Google Chrome 14

Chrome 14 发布于 2011 年 9 月 16 日。

新特性：

- 加强安全的网络协议（**DNSSEC** 上的 **HTTPS**）
- 更新 **JavaScript** 引擎 到 V8 3.4.3.0
- 打印预览（**Mac**）

Google Chrome 13

Chrome 13 发布于 2011 年 8 月 2 日。

新特性：

- 即时页面**Instant Pages** - 更快的谷歌搜索结果（预渲染网页）
- 打印预览（**Windows/Linux**）

Google Chrome 12

Chrome 12 发布于 2011 年 6 月 7 日。

新特性：

- 更安全
- 支持硬件加速的 3D CSS

Google Chrome 11

Chrome 11 发布于 2011 年 4 月 27 日。

新特性：

- 品牌新的 logo
- 增加 HTML 语音（Chrome 可以把您的语音转换为文本）

Google Chrome 10

Chrome 10 发布于 2011 年 3 月 8 日。

新特性：

- 更快
- 新的密码同步功能
- 新的浏览器设置

Google Chrome 9

Chrome 9 发布于 2011 年 2 月 3 日。

新特性：

- Chrome 网络商店（Chrome Web Store）
- 3D 图形的 WebGL
- Chrome 即时（Chrome Instant） - 一旦您开始输入，就开始加重您经常访问的网页。

Google Chrome 8

Chrome 8 发布于 2010 年 12 月 2 日。

新特性：

- 稳定性的改进
- 更好的 CSS3 支持
- 更好的 HTML5 支持

Google Chrome 7

Chrome 7 发布于 2010 年 10 月 21 日。

新特性：

- 修复大量的 bug
- 新的 HTML5 解析器

Google Chrome 6

Chrome 6 发布于 2010 年 9 月 2 日。

新特性：

- Chrome 同步：如果您已经登录到 Chrome，那么您的浏览器设置的任何改动都会被同步到谷歌帐户。

Google Chrome 5

Chrome 5 发布于 2010 年 5 月 21 日。

新特性：

- 改进的 JavaScript 性能
- 增加对 HTML5 的支持（地理定位、应用程序缓存、Web sockets、拖放）
- 集成的 Adobe Flash Player
- 针对 Mac 和 Linux 用户，Chrome 5 是最后的测试
- 让您在隐身模式浏览时也能使用扩展库中的功能

Google Chrome 4

Chrome 4 发布于 2010 年 1 月 25 日。

新特性：

- 更快，更安全
- 增加了对 HTML5 的支持
- 在 Chrome 扩展库中增加了成千上万个新的工具
- 提供了多台电脑之间的书签同步

Google Chrome 3

Chrome 3 发布于 2009 年 10 月 12 日。

新特性：

- 重新设计的起始页（新标签页）
- JavaScript 执行速度比上一个版本提升了 25%
- 改进的地址栏
- 支持 HTML5 <video> 和 <audio>
- 为浏览器设计了不同的主题

Google Chrome 2

Chrome 2 发布于 2009 年 5 月 24 日。

新特性：

- 新的标签屏幕（Tab Screen），其中显示您最经常访问的网页，您也可以把不想显示在标签屏幕（Tab Screen）中的页面删除
- 自动填充表单
- 全屏模式
- 修复了超过 300 个 bug
- JavaScript 执行速度比上一个版本提升了 35%

Google Chrome 1

Chrome 1 发布于 2008 年 12 月 11 日。

第一个稳定版本！

Google 花费了 100 天的时间将 Beta 版本提升至正式版。在这 100 天中，Google 完善了该浏览器，并搜集了来自用户的需求。

Google Chrome 的第一个 Beta 版本

Chrome 的第一个 Beta 1 版本发布于 2008 年 9 月 2 日。

苹果 Safari 浏览器

Safari 浏览器



在 2003 年 1 月，史蒂夫乔布斯（Steve Jobs）宣布苹果正在开发自己的浏览器：Safari。

在此之前，Mac 系统使用 Netscape Navigator 或 Internet Explorer 作为其默认浏览器。

第一个正式的 ("out-of-beta") Safari 版本于 2003 年 6 月发布。在 2005 年 4 月，Safari 成为 Mac 系统的默认浏览器。

如同苹果的许多产品，Safari 以易用和清爽的设计闻名。Safari 支持 Mac 和 Windows 系统。

下载 Safari

Safari 统计

下表是 浏览器统计信息 中关于 Safari 使用情况的细节：

2014	总计	S 7	S 6	S 5
5 月	3.8 %	3.0 %	0.7 %	0.3 %
4 月	4.0 %	2.8 %	0.8 %	0.4 %
3 月	3.9 %	2.5 %	0.9 %	0.5 %
2 月	4.0 %	2.5 %	1.0 %	0.5 %
1 月	3.9 %	2.3 %	1.1 %	0.5 %

2013	总计	S 7	S 6	S 5	S 4
12 月	3.8 %	2.2 %	1.1 %	0.5 %	0.0 %
11 月	4.0 %	2.0 %	1.4 %	0.6 %	0.0 %
10 月	3.8 %	1.0 %	2.1 %	0.7 %	0.0 %
9 月	3.9 %	3.1 %	0.8 %	0.0 %	
8 月	3.9 %	3.0 %	0.9 %	0.0 %	
7 月	3.6 %	2.8 %	0.8 %	0.0 %	
6 月	3.9 %	3.0 %	0.9 %	0.0 %	
5 月	4.0 %	2.9 %	1.0 %	0.0 %	
4 月	4.0 %	2.8 %	1.1 %	0.0 %	
3 月	4.1 %	2.8 %	1.2 %	0.1 %	
2 月	4.1 %	2.7 %	1.3 %	0.1 %	
1 月	4.2 %	2.7 %	1.4 %	0.1 %	

2012	总计	S 6	S 5	S 4
12 月	4.2 %	2.6 %	1.5 %	0.1 %
11 月	4.4 %	2.6 %	1.7 %	0.1 %
10 月	4.3 %	2.3 %	1.9 %	0.1 %
9 月	4.2 %	1.9 %	2.2 %	0.1 %
8 月	4.0 %	1.4 %	2.5 %	0.1 %
7 月	3.9 %	0.2 %	3.6 %	0.1 %
6 月	4.1 %	4.0 %	0.1 %	
5 月	4.3 %	4.2 %	0.1 %	
4 月	4.5 %	4.4 %	0.1 %	
3 月	4.4 %	4.3 %	0.1 %	
2 月	4.5 %	4.4 %	0.1 %	
1 月	4.3 %	4.2 %	0.1 %	

2011	总计	S 5	S 4	S 3
12 月	4.2 %	4.1 %	0.1 %	0.0 %
11 月	4.2 %	4.1 %	0.1 %	0.0 %
10 月	4.2 %	4.0 %	0.2 %	0.0 %
9 月	4.0 %	3.8 %	0.2 %	0.0 %
8 月	3.8 %	3.6 %	0.2 %	0.0 %
7 月	3.6 %	3.4 %	0.2 %	0.0 %
6 月	3.7 %	3.5 %	0.2 %	0.0 %
5 月	4.0 %	3.6 %	0.3 %	0.1 %
4 月	4.1 %	3.7 %	0.3 %	0.1 %
3 月	4.0 %	3.6 %	0.3 %	0.1 %
2 月	4.1 %	3.6 %	0.4 %	0.1 %
1 月	4.0 %	3.5 %	0.4 %	0.1 %

2010	总计	S 5	S 4	S 3
12 月	3.8 %	3.2 %	0.5 %	0.1 %
11 月	4.0 %	3.1 %	0.7 %	0.2 %
10 月	3.9 %	3.1 %	0.7 %	0.1 %
9 月	3.7 %	2.9 %	0.7 %	0.1 %
8 月	3.5 %	2.6 %	0.8 %	0.1 %
7 月	3.4 %	2.3 %	1.0 %	0.1 %
6 月	3.6 %	1.4 %	2.1 %	0.1 %
5 月	3.5 %	3.3 %	0.2 %	
4 月	3.7 %	3.5 %	0.2 %	
3 月	3.7 %	3.5 %	0.2 %	
2 月	3.8 %	3.5 %	0.3 %	
1 月	3.7 %	3.4 %	0.3 %	

2009	总计	S 4	S 3
12 月	3.6 %	3.3 %	0.3 %
11 月	3.8 %	3.4 %	0.4 %
10 月	3.8 %	3.3 %	0.5 %
9 月	3.6 %	3.0 %	0.6 %
8 月	3.3 %	2.6 %	0.7 %
7 月	3.3 %	2.4 %	0.9 %
6 月	3.1 %	1.7 %	1.4 %
5 月	3.0 %	0.9 %	2.1 %
4 月	3.0 %	0.9 %	2.1 %
3 月	3.1 %	1.0 %	2.1 %
2 月	3.0 %	0.2 %	2.8 %
1 月	3.0 %	0.1 %	2.9 %

2008	总计	S 3
12 月	2.7 %	2.7 %
11 月	2.7 %	2.7 %
10 月	2.8 %	2.8 %
9 月	2.7 %	2.7 %
8 月	2.6 %	2.6 %
7 月	2.5 %	2.5 %
6 月	2.6 %	2.6 %
5 月	2.4 %	2.4 %

以上统计数据是基于 W3CSchool 网站的用户。

Safari 7

Safari 7/6.1 于 2013 年 6 月 10 日发布。

Safari 7（针对 OS X Mavericks）和 Safari 6.1（针对 Lion 和 Mountain Lion）随着 OS X Mavericks 于 2013 年 10 月 22 日发布。

新特性：

- 改进的 **JavaScript** 性能和内存使用

- **Top** 站点的新外观和侧边栏（**Sidebar**）
- **iCloud Keychain** - 存储/生成在线使用的随机密码。也可以安全地存储您的信用卡信息
- 分享链接的功能 - 收集了所有的分享链接
- 电源保护功能 - 在不使用插件时，暂停这些插件

Safari 6

Safari 6 于 2012 年 7 月 25 日发布。

苹果已经声明："Safari 6 可以在 Mountain Lion 和 Lion 上运行。Safari 5 可继续在 Windows 上运行。"

新特性：

- **iCloud** 标签
- **Web audio API** - 允许用户在交互式的 web 应用中创建音频效果
- 支持 **CSS** 过滤器
- 更好的 **HTML5** 支持 - 定时文本的轨道，媒体同步
- 改进的 **JavaScript** 支持 - ECMA 262 版本 5.1
- 浏览权限检测
- 重新设计的 **Web inspector**
- 自定义阅读

Safari 6 集成到 OS X Mountain Lion - 不能从苹果网站或其他资源站下载。

苹果通过 Software Update 为 OS X Lion 的用户发布 Safari 6。苹果不再发布 Lion 之前的 OS X 版本和 Windows 版本。

Safari 5

Safari 5 于 2010 年 6 月 7 日发布。

Safari 5 可以在 Mac 和 Windows 上运行。

新特性：

- **Nitro JavaScript** 引擎 **-** Mac 版本的 Safari 5 由 Nitro JavaScript 引擎驱动，执行 JavaScript 的速度比 Safari 4 快 30%。
- 使用 **DNS** 预取来加速页面加载 **-** 就像 Google 的 Chrome 浏览器，利用 DNS（域名系统）预获取功能提高了加载新网页的速度，同时改进了历史页面缓存，令返回到这些页面的响应更加快速。
- **Safari** 阅读器 **-** 通过新的可滚动显示的界面呈现文章，排除了多余的内容和混乱的信息，不论单页还是多页文章都将变得更易阅读。当 Safari 5 检测到某篇文章时，用户可

以点击智能地址栏中的阅读器图标以显示整篇文章，获得清晰、连续不间断的阅读界面，并能够将其放大、打印或通过电子邮件发送。

- **HTML5** 特性 - 新增了十多种强大的 HTML5 特性，例如 HTML5 视频的全屏回放和对隐藏式字幕的支持，可让网页开发者创造出富媒体体验。Safari 5 新增的其它 HTML5 特性包括 HTML5 地点标记、HTML5 页面分割元素、HTML5 拖拽属性、HTML5 表单验证、HTML5 Ruby、HTML5 AJAX 历史、EventSource 和 WebSocket 等。
- 安全扩展 - 新的免费的 Safari 开发者计划允许开发者使用基于标准 web 技术（例如 HTML5、CSS3 和 JavaScript）的扩展工具来自定义和增强 Safari 5。Safari 5 新增的扩展创建器简化了扩展工具的开发、安装和打包工作。为了增强安全性和稳定性，Safari 的扩展工具均运行在沙盒中，使用 Apple 的数字证书进行签名，并且只运行于浏览器中。
- 内建 **Bing** 搜索 - 可选择 Google、Yahoo! 或 Bing 作为 Safari 搜索栏的搜索服务。

更旧的 **Safari** 版本

Safari 4（针对 **Mac** 和 **Windows**） - 于 2009 年 6 月发布。

Safari 3（针对 **Mac** 和 **Windows**） - 于 2007 年 10 月发布。

Safari 2（针对 **Mac**） - 于 2005 年 4 月发布。

Safari 1（针对 **Mac**） - 于 2003 年 6 月发布。

Opera 浏览器

Opera 浏览器



Opera 浏览器是免费的，它是目前最小最快的浏览器！

Opera 是作为挪威电信公司 Telenor 的一个研究项目于 1994 年启动的，并于 1995 发展为一个独立的开发公司，Opera Software ASA。

Opera Software ASA 研发了 Opera web 浏览器，是桌面和设备市场的 web 浏览器开发领域的业界领导者。

Opera 浏览器由于其较之其他浏览器具有更快、更小且标准兼容性更强的优点，已经得到来自终端用户和业界的国际性赞誉。

下载Opera

Opera 统计

下表是 浏览器统计信息 中关于 Opera 使用情况的细节：

2014	总计	O 20	O 19	O 18	O 12	O Mini	其他
5 月	1.8 %	0.3 %	0.0 %	0.0 %	0.4 %	0.7 %	0.4 %
4 月	1.8 %	0.6 %	0.0 %	0.0 %	0.4 %	0.8 %	0.0 %
3 月	1.8 %	0.5 %	0.1 %	0.0 %	0.5 %	0.7 %	0.0 %
2 月	1.9 %	0.5 %	0.1 %	0.5 %	0.7 %	0.1 %	
1 月	1.8 %	0.4 %	0.5 %	0.7 %	0.2 %		

2013	总计	O 18	O 17	O 16	O 15	O 12	O 11	O Mini	其他
12 月	1.9 %	0.4 %	0.1 %	0.0 %	0.0 %	0.5 %	0.0 %	0.7 %	0.2 %
11 月	1.8 %	0.2 %	0.2 %	0.0 %	0.0 %	0.6 %	0.0 %	0.7 %	0.1 %
10 月	1.7 %	0.2 %	0.1 %	0.0 %	0.6 %	0.0 %	0.6 %	0.2 %	
9 月	1.7 %	0.9 %	0.0 %	0.8 %	0.0 %				
8 月	1.8 %	0.7 %	0.1 %	0.7 %	0.3 %				
7 月	1.6 %	0.8 %	0.1 %	0.7 %	0.0 %				
6 月	1.7 %	1.0 %	0.1 %	0.6 %	0.0 %				
5 月	1.6 %	1.0 %	0.1 %	0.6 %	0.0 %				
4 月	1.7 %	1.0 %	0.1 %	0.5 %	0.1 %				
3 月	1.8 %	1.1 %	0.1 %	0.6 %	0.0 %				
2 月	1.8 %	1.1 %	0.1 %	0.5 %	0.1 %				
1 月	1.9 %	1.1 %	0.1 %	0.6 %	0.1 %				

2012	总计	O 12	O 11	O 10	O Mini	其他
12 月	2.1 %	1.2 %	0.1 %	0.0 %	0.6 %	0.2 %
11 月	2.0 %	1.3 %	0.1 %	0.0 %	0.5 %	0.1 %
10 月	2.0 %	1.2 %	0.2 %	0.0 %	0.5 %	0.1 %
9 月	2.1 %	1.2 %	0.2 %	0.0 %	0.6 %	0.1 %
8 月	2.2 %	1.2 %	0.3 %	0.0 %	0.7 %	0.0 %
7 月	2.1 %	1.1 %	0.4 %	0.0 %	0.6 %	0.0 %
6 月	2.2 %	0.5 %	1.1 %	0.0 %	0.6 %	0.0 %
5 月	2.2 %	0.1 %	1.5 %	0.1 %	0.5 %	0.0 %
4 月	2.3 %	0.1 %	1.6 %	0.1 %	0.5 %	0.0 %
3 月	2.3 %	0.1 %	1.6 %	0.1 %	0.5 %	0.0 %
2 月	2.3 %	0.1 %	1.6 %	0.1 %	0.5 %	0.0 %
1 月	2.4 %	0.1 %	1.7 %	0.1 %	0.5 %	0.0 %

2011	总计	O 11	O 10	O Mini	其他
12 月	2.5 %	1.8 %	0.1 %	0.5 %	0.1 %
11 月	2.4 %	1.8 %	0.1 %	0.5 %	0.0 %
10 月	2.4 %	1.8 %	0.1 %	0.5 %	0.0 %
9 月	2.2 %	1.7 %	0.1 %	0.4 %	0.0 %
8 月	2.3 %	1.7 %	0.1 %	0.4 %	0.1 %
7 月	2.4 %	1.7 %	0.1 %	0.5 %	0.1 %
6 月	2.4 %	1.7 %	0.2 %	0.4 %	0.1 %
5 月	2.4 %	1.8 %	0.2 %	0.4 %	0.0 %
4 月	2.6 %	1.9 %	0.2 %	0.4 %	0.1 %
3 月	2.5 %	1.9 %	0.2 %	0.3 %	0.1 %
2 月	2.5 %	1.8 %	0.3 %	0.3 %	0.1 %
1 月	2.5 %	1.7 %	0.4 %	0.3 %	0.1 %

2010	总计	O 11	O 10	O 9.5	其他
12 月	2.2 %	0.7 %	1.4 %	0.1 %	0.0 %
11 月	2.3 %	0.1 %	2.1 %	0.1 %	0.0 %
10 月	2.2 %	2.1 %	0.1 %	0.0 %	
9 月	2.2 %	2.0 %	0.1 %	0.1 %	
8 月	2.3 %	2.1 %	0.1 %	0.1 %	
7 月	2.3 %	2.1 %	0.2 %	0.0 %	
6 月	2.1 %	1.9 %	0.2 %	0.0 %	
5 月	2.2 %	2.0 %	0.2 %	0.0 %	
4 月	2.2 %	2.0 %	0.2 %	0.0 %	
3 月	2.2 %	1.9 %	0.3 %	0.0 %	
2 月	2.1 %	1.8 %	0.3 %	0.0 %	
1 月	2.2 %	1.8 %	0.4 %	0.0 %	

2009	总计	O 10	O 9.5	O 9	其他
12 月	2.3 %	1.8 %	0.4 %	0.1 %	0.0 %
11 月	2.3 %	1.7 %	0.5 %	0.1 %	0.0 %
10 月	2.3 %	1.5 %	0.7 %	0.1 %	0.0 %
9 月	2.2 %	1.2 %	0.9 %	0.1 %	0.0 %
8 月	2.1 %	0.4 %	1.6 %	0.1 %	0.0 %
7 月	2.1 %	0.3 %	1.7 %	0.1 %	0.0 %
6 月	2.1 %	0.2 %	1.7 %	0.1 %	0.1 %
5 月	2.2 %	0.1 %	1.9 %	0.1 %	0.1 %
4 月	2.2 %	0.1 %	1.9 %	0.1 %	0.1 %
3 月	2.3 %	0.1 %	1.9 %	0.2 %	0.1 %
2 月	2.2 %	0.1 %	1.9 %	0.2 %	0.0 %
1 月	2.3 %	0.1 %	1.9 %	0.2 %	0.1 %

2008	总计	O 10	O 9.5	O 9	其他
12 月	2.4 %	0.1 %	2.0 %	0.2 %	0.1 %
11 月	2.3 %	1.9 %	0.3 %	0.1 %	
10 月	2.2 %	1.7 %	0.4 %	0.1 %	
9 月	2.0 %	1.5 %	0.4 %	0.1 %	
8 月	2.1 %	1.5 %	0.5 %	0.1 %	
7 月	1.9 %	1.3 %	0.6 %	0.0 %	
6 月	1.7 %	1.0 %	0.6 %	0.1 %	
5 月	1.5 %	0.1 %	1.3 %	0.1 %	

以上统计数据是基于 W3CSchool 网站的用户。

Opera 不仅免费而且符合 Web 标准！

Opera 浏览器是免费的！早期的旗帜广告和许可费制度已经被撤销了！

Opera 浏览器具有更快、更小且标准兼容性性更强的优点。

如果您的网站在 Opera 中运行良好，那么可以确定它符合标准！只需要用符合标准的代码来编写您的页面，就可确保您的网站运行良好，不论是在所有主流浏览器中，还是在所有主流平台以及操作系统上。

Opera 支持所有在用的 Web 标准，诸如 CSS、HTML、XHTML、HTTP、DOM、XML、XSL、ECMAScript（JavaScript）、PNG、WML、SVG、Unicode、Unicode Bidirectional Algorithm 等等。

Opera 20

Opera 20 于 2014 年 3 月 4 日发布，包括更新到最新的 Chromium 版本 33。

Opera 20 可以在 Mac 和 Windows 上运行。

新特性：

- 可拖拽的书签 - 拖拽您的标签到书签栏，可以保存这些标签。拖拽您的书签到快速拨号（Speed Dial），可以创建新的条目。
- 新的快速拨号（**Speed Dial**）条目 - 在高级设置中，可以改变快速拨号（Speed Dial）条目的外观。

Opera 19

Opera 19 于 2014 年 1 月 28 日发布，包括更新到最新的 Chromium 版本 32。

Opera 19 可以在 Mac 和 Windows 上运行。

新特性：

- 书签栏 - 可以保存标签到书签栏，可以很方便地管理书签栏里边的标签，还可以很容易地点击书签栏中标签来访问网页。
- 主题创建 - 上传图像或者使用来自网络上的图像来自定义浏览器的外观。

Opera 18

Opera 18 于 2013 年 11 月 19 日发布，包括更新到最新的 Chromium 版本 31。

Opera 18 可以在 Mac 和 Windows 上运行。

新特性：

- 媒体访问 - 允许网站访问您的相机和麦克风，包括对 HTML5 媒体访问的支持。
- 主题 - 可以通过主题管理器安装来自 addons.opera.com 上的主题。
- 摇杆（**Rocker**）手势 - 通过在鼠标按钮上移动您的手指，来向前/向后导航标签的历史。
- 增强标签的功能 - 您可以在打开的窗口之间拖拽标签。

Opera 17

Opera 17 于 2013 年 10 月 10 日发布，包括更新到最新的 Chromium 版本 30。

Opera 17 可以在 Mac 和 Windows 上运行。

新特性：

- 固定标签 - 允许您固定标签到标签栏上（有助于防止您意外关闭标签）。
- 自定义搜索 - 您可以添加自定义搜索引擎到联合搜索和地址栏。

Opera 16

Opera 16 于 2013 年 8 月 27 日发布，包括更新到最新的 Chromium 版本 29。

Opera 16 可以在 Mac 和 Windows 上运行。

新特性：

- 自动填充 - 您可以安全地存储在线使用的地址、电话号码、电子邮件地址和信用卡。
- 地理位置 - 您可以通过网站分享您的位置信息。

Opera 15

Opera 15 于 2013 年 7 月 2 日发布。

Opera 15 可以在 Mac 和 Windows 上运行。

Opera 15 是 Opera 的一个重大的再造。浏览器的源代码是基于 Chromium 和 Blink，以及 WebKit 渲染引擎。这个版本的浏览器的一切都是新的：

- 新的渲染引擎，基于 WebKit
- 新的用户界面
- 新的能力
- 新的特性
- 改进的站点兼容性
- 自动更新

新特性：

- 组合的地址与搜索栏
- 增强的快速拨号（**Speed Dial**） - 允许您将快速拨号条目分组。
- 储备特性 - 记录位置、元数据、网页截图，并把这些信息放置在浏览器起始页上的可折叠列表项中，可以通过关键字搜索，也可以通过截图进行可视化浏览。
- 获得知识 - 显示新闻和专题内容，可以通过类别或位置/语言进行过滤。
- 越野（**Off-Road**）模式 - 采用 Opera Mobile 所使用的服务器端压缩技术。

Opera 12

Opera 12 于 2012 年 6 月 13 日发布。

Opera 12 可以在 Linux、Mac、FreeBSD 和 Windows 上运行。

新特性：

- 改进的速度
- 支持拍照 - 第一个支持 W3C getUserMedia 规范的浏览器。
- 主题 - Opera 12 引进了轻量级的主题。
- 更好的安全徽章
- 改进对 **HTML5** 和 **CSS3** 的支持 - 比如 HTML5 拖放、CSS3 过渡和动画。
- 新增 **5** 种新的语言 - 阿拉伯语（Arabic）、波斯语（Persian）、乌尔都语（Urdu）、希伯莱语（Hebrew）和哈萨克语（Kazakh）。
- 一个页面缩放滑块
- 支持不跟踪

Opera 11.60

Opera 11.60 于 2011 年 12 月 6 日发布。

Opera 11.60 可以在 Linux、Mac、FreeBSD 和 Windows 上运行。

新特性：

- 新的浏览器引擎 - 更快和更稳定的互联网体验。
- 新的邮件设计 - 更简洁的布局，消息分组，更方便的导航，在收件箱中更直观的视图。
- 修改地址字段 - 只需要按下地址栏中的星号，即可添加页面到书签或者快速拨号（Speed Dial）中
- 改进对 **HTML5**、**CSS3**、**SVG** 的支持。

Opera 11.50

Opera 11.50 于 2011 年 6 月 28 日发布。

Opera 11.50 可以在 Linux、Mac、FreeBSD 和 Windows 上运行。

新特性：

- 时尚的用户界面
- 为 **Opera** 的快速拨号（**Speed Dial**）特性添加功能 - 通过扩展嵌入您的快速拨号（Speed Dial），让您可以即时更新，而不需要通过添加缩略图链接到收藏夹中。
- 密码同步 **-** 让您与其他 Opera 浏览器安全地同步密码。
- 更快的图形性能
- 改进对 **HTML5**、**CSS3**、**SVG** 的支持

Opera 11.10

Opera 11.10 于 2011 年 4 月发布。

Opera 11.10 可以在 Linux、Mac、FreeBSD 和 Windows 上运行。

新特性：

- **Speed Dial 2.0** - 可以更清晰地预览您喜爱的网页，并且可以动态显示网站最新的内容。快速拨号项的使用数量不做限制。
- 增**强**的 **Opera Turbo** - 在拥挤的 Wi-Fi 热点地区，或者在使用移动电话或拨号连接时，提高您的浏览速度。
- 简单的插件安装 - 在这个版本中，最流行的插件，Adobe Flash Player，可以无缝自动安装。
- 新的 **CSS3** 支持 - Opera 现在支持 CSS3 线性渐变和多列。

Opera 11

Opera 11 于 2010 年 12 月发布。

Opera 11 可以在 Linux、Mac、FreeBSD 和 Windows 上运行。

新特性：

- **Opera Presto 2.7** 渲染引擎 - 更好地支持 CSS3 和 HTML5。
- 标签堆叠 - 我们经常打开许多标签，以致于难以调整和管理。Opera 11 让您可以拖动一个标签到另一个标签上，用来创建标签组。限制您可以让打开的几十个网页有组织地显示，且您可以很方便地进行调整和管理。
- 安全地址字段 - 一种改进的地址字段，使得它更容易在网络上保持安全，并且它隐藏了长网络地址的复杂性。同时它还提供了更安全的信息来帮助您在浏览时保持安全，只需要点击网站的徽章就能看到您所访问的网站的安全信息。
- 视觉鼠标手势（**Visual Mouse Gestures**） - 通过鼠标的快速轻弹，即可完成常用的浏览器动作。在网页上按住鼠标右键，会显示一个指示如何执行可用手势的视觉向导。
- 针对微软 **Windows** 平台的 **Opera** 安装程序 - Opera 有一个新的安装程序，比之前使用的 MSI 安装程序快多了。
- 新的扩展 **&** 邮件面板 - 一堆新的扩展，一个新的右键面板，以便更加快速地使用电子邮件。

更旧的 Opera 版本

Opera 10.5 - 于 2010 年 3 月发布。

Opera 10.0 - 于 2009 年 9 月发布。

Opera 9.5 - 于 2008 年 6 月发布。

Opera 9.0 - 于 2006 年 6 月发布。

Opera 8.5 - 于 2005 年 9 月发布。

Opera 8.0 - 于 2005 年 4 月发布。

Mozilla 项目

什么是 Mozilla？



Mozilla 不是一款 web 浏览器！

Mozilla 是一个使用诸如 CSS、XML、RDF 等 web 标准来构建 web 应用程序的框架。

Mozilla 是一项开发用在 Mozilla 应用套件中的程序代码的非营利性的开源 web 开发项目。

Mozilla 应用程序套件是一套完整的 web 应用程序（浏览器、聊天客户端、新闻客户端、邮件客户端等等）。

Mozilla 认为，互联网是一个需要改善和保护的公共资源。

Mozilla 的产品

- **Firefox** - 当今最流行的互联网浏览器之一
- **Thunderbird** - 一个电子邮件和新闻组客户端，具有安全、快速、易用的特点
- **SeaMonkey** - - 集浏览、收发电子邮件、聊天和编辑于一体
- **Bugzilla** - bug 跟踪工具
- **Camino** - 一款用于 Mac 的 web 浏览器
- **Lightning & Sunbird** - 日历的扩展和应用
- **Composer** - 网页编辑器

产品可通过下面的地址来下载： http://www.mozilla.org

Mozilla 项目的历史

1998 年，随着 Netscape 浏览器源代码的发布，Mozilla 项目作为一个开源社区被创建。

一年内，来自世界各地的新的社区成员已经为 Netscape 的下一个浏览器增加了许多新的功能，并增强了它现有的功能，同时 Mozilla 项目也逐渐发展壮大。成员们不再只是致力于 Netscape 的下一个浏览器，而是开始创建各种浏览器、开发工具和其他项目。

2002 年，第一个主要版本 Mozilla 1.0 发布了。这个套件对浏览器、电子邮件客户端和其他应用程序做了很多改进。但并没有很多人使用它（超过 90% 的互联网用户使用 Internet Explorer）。同年，Mozilla 发布了 Phoenix（后来重命名为 Firefox）的第一个版本。

2003 年，Mozilla 项目创建 Mozilla 基金会，这是一个独立的非营利组织。Mozilla 基金会持续管理 Mozilla 项目的日常运作。

2004 年，发布了 Firefox 1.0，这是一个重大的成功。在不超过一年的时间内，Firefox 下载量超过 100 百万次。Firefox 的受欢迎度有助于把选择器交还给用户。

2008 年，Firefox 达到 20% 的全球市场份额。

2008 年，Mozilla 庆祝它成立十周年。十年来，社区已经表明，商业公司可以通过开源项目的合作来获取利益。

Mozilla 基金会

Mozilla 基金会创建于 2003 年 7 月，位于美国加利福尼亚州的芒廷维尤。

Mozilla 基金会简称 Mozilla（缩写 MF 或 MoFo），是为支持和领导开源的 Mozilla 项目而设立的一个非营利组织。该组织制定管理开发政策，经营关键基础组织并管理商标及其他知识产权。它拥有一个称作 Mozilla 公司的子公司，雇佣了一些 Mozilla 开发人员并协调 Mozilla Firefox 网页浏览器以及 Mozilla Thunderbird 电子邮件客户端的发行版。

Mozilla 基金会把自己描述为"一个致力于在互联网领域提供多样化选择和创新的公益组织"。

在最初阶段，Mozilla 基金会开始涉足比 mozilla org 更广的领域，把以前推给 Netscape 和 Mozilla 合作伙伴的事情都拿来做了。

在向"面向最终用户"的转型举动中，Mozilla 基金会和一些商业公司签约来售卖包含 Mozilla 软件的光盘并且提供电话支持服务。在这些举动中，Mozilla 基金会选择了以前 Netscape 的供应商。

Mozilla 基金会变得对自己的知识产权更加的自信，他们推出了自己商标使用的新政策。

Mozilla 基金会也开始了市场拓展等的新项目。

随着 Mozilla 公司的成立，Mozilla 基金会把所有的软件开发和商业相关的活动都转移给了这个新的下属机构。

Mozilla 基金会现在只专注于监管和战略等事宜，它也继续管理一些没有产品化的项目，比如 Camino 和 SeaMonkey。

Mozilla 基金会现在拥有 Mozilla 商标和其他知识产权，并且全部授权 Mozilla 公司使用。

Mozilla 基金会还控制着 Mozilla 的程序源代码库并决定着谁可以提交代码入库。

Mozilla 基金会的路线图

Mozilla 决定制定一个新的开发路线图。下面列出了新路线图中的一些要点：

- 专注于独立的应用程序（FireFox 浏览器、Thunderbird 邮件/新闻应用程序以及独立的设计器）
- 使 Firefox 和 Thunderbird 成为 Mozilla 的首要产品
- 用大型的 Mozilla 部署来维护针对企业和组织的SeaMonkey（现在的 Mozilla 浏览器）应用程序套件
- 使用一年的研发周期把 Mozilla 1.4 分支作为由组织使用的 "distributor/vendor" 分支来进行维护
- 修复至关重要的 Gecko layout architecture 的漏洞。所有的 Mozilla 应用程序都得益于这些 Gecko 的改进。
- 精益求精。做得更少，但要做得更好！

令人混淆的 Mozilla 名称

第一款 Netscape 浏览器使用了名为 Mozilla 的代码引擎。*Netscape 1.0* 是依靠名为 *Mozilla 1.0* 的代码引擎来驱动的。Netscape 2.0 使用 Mozilla 2.0，Netscape 3.0 使用 Mozilla 3.0，而 Netscape 4.0 使用了 Mozilla 4.0。

在 1998 年，Netscape 4 将其源代码公开 - 同时把 Netscape 5 的开发确立为一个开源项目。

这个创建 Netscape 5 的开源项目被称为 *"The Mozilla Project"*。奇怪的是，这个 Mozilla 项目的代码引擎被称为 *Gecko*。

不幸的是，在 4.0 发布之后，对下一代浏览器的研发耗费了 Netscape 超过三年半的时间。这次延迟破坏了 Netscape 作为微软 IE 浏览器的可靠的备选方案的可能性。就在 Mozilla 项目启动不久，微软就发布了它的 IE 5.0，而在 Netscape 设法发布一款可工作的浏览器之前，微软的 IE 6.0 也就绪了。

基于 *Gecko M18 (Milestone 18)* 的 *Netscape 6.0* 于 2000 年 11 月发布。

在 Netscape 6.0 发布之后，Mozilla 项目开始研发基于名为 *Gecko 1.0* 引擎的 *Netscape 7*。

Netscape 6 和 7 均构建于 Mozilla 之上，Netscape 和 Mozilla 是几乎相同的应用程序套件。

Netscape 7 声称其使用了名为 *Gecko 1.0* 的代码引擎。

此刻，Mozilla 项目正在开发一款名为 *Firefox* 的新浏览器。在过去，Firefox 被称为 *Mozilla Firebird*（而 Mozilla Firebird 过去被称为 *Phoenix*，它声称是 *Mozilla* 的一个新版本）。

Netscape 浏览器

Netscape 已宣告死亡



Netscape 是首个商业化的 web 浏览器。它发布于 1994 年。Netscape 虽是一个商业软件，但它也提供了可在 Unix、VMS、Macs 和 Microsoft Windows 等操作系统上运行的免费版本。

1997 年 4 月，微软开始推出 IE 4.0 系列版本，这个版本几乎是恶评如潮，同样发展到 4.0 的 Netscape 却一路高歌猛进。然而，Netscape 的好日子并没有一直延续下去。到了 Windows 98 时代，微软公司拿出一记必胜绝招，将 IE 与 Windows 98 捆绑在一起免费提供，此时的 IE 也慢慢变得好了起来。既然操作系统中提供了浏览器，为什么还要单独购买一个呢？绝大多数用户都有这样的想法，结果在很短的时间内，Netscape 的用户群体迅速萎缩，市场占有率急剧下降。在 IE 的竞争下，Netscape 逐渐丧失了它的市场份额。

面对微软的强大威胁，Netscape 在 1998 年 11 月决定将软件免费、且公开所有的程序源码，把主要力量放在商业市场。这个措施几乎还没来得及实施， Netscape 就被 AOL 收购。**1998 年，AOL 以 42** 亿美元收购了 **Netscape**。2003 年，Mozilla 基金会与 Netscape 分离开，作为它的前母公司的 AOL 提供了 200 万美元作为分手礼物。

2007 年 **12** 月 **28** 日：**Netscape** 开发者宣布由于市场份额低，**AOL**（美国在线服务公司）将在 **2008** 年 **2** 月 **1** 日停止开发他们的 **web** 浏览器。

Netscape 的开发总监，**Tom Drapeau** 写道：**"AOL** 把重点转向广告网络业务，导致没有更多的资金投入到研发中，促使 **Netscape** 浏览器达不到用户心目中的期望值。**"**

扩展阅读：http://blog.netscape.com/2007/12/28/end-of-support-for-netscape

Netscape 的问题

Netscape 的 4.x 系列的浏览器对 CSS 的支持很差，也不支持 XML。

在 4.0 之后，Netscape 花了将近三年半的时间研发其下一代的浏览器。这次延迟明显地减少了 Netscape 垄断浏览器市场的可能性。

Netscape 版本

Netscape Navigator 9 - 发布于 2007。

Netscape 8 - 发布于 2006。

Netscape 7 - 发布于 2002。

Netscape 6 - 发布于 2000。支持 CSS 和 XML。

Netscape 5 - Netscape 跳过版本 5。

Netscape Communicator 4 - 发布于 1997。

Netscape Navigator 3 - 发布于 1996。

Netscape Navigator 2 - 发布于 1996。

Netscape 1 - 发布于 1994。

HTTP教程

HTTP 简介

HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。。

HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP 工作原理

HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。

Web服务器有：Apache服务器，IIS服务器（Internet Information Services）等。

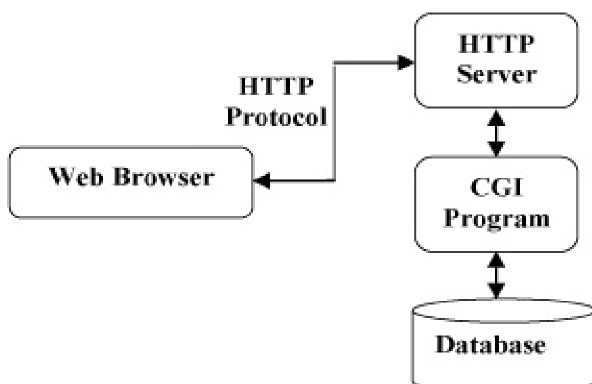
Web服务器根据接收到的请求后，向客户端发送响应信息。

HTTP默认端口号为80，但是你也可以改为8080或者其他端口。

HTTP三点注意事项：

- HTTP是无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- HTTP是媒体独立的：这意味着，只要客户端和服务器知道如何处理的数据内容，任何类型的数据都可以通过HTTP发送。客户端以及服务器指定使用适合的MIME-type内容类型。
- HTTP是无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

以下图表展示了HTTP协议通信流程：



HTTP 消息结构

HTTP是基于客户端/服务端（C/S）的架构模型，通过一个可靠的链接来交换信息，是一个无状态的请求/响应协议。

一个HTTP"客户端"是一个应用程序（Web浏览器或其他任何客户端），通过连接到服务器达到向服务器发送一个或多个HTTP的请求的目的。

一个HTTP"服务器"同样也是一个应用程序（通常是一个Web服务，如Apache Web服务器或IIS服务器等），通过接收客户端的请求并向客户端发送HTTP响应数据。

HTTP使用统一资源标识符（Uniform Resource Identifiers, URI）来传输数据和建立连接。

一旦建立连接后，数据消息就通过类似Internet邮件所使用的格式[RFC5322]和多用途Internet邮件扩展（MIME）[RFC2045]来传送。

客户端请求消息

客户端发送一个HTTP请求到服务器的请求消息包括以下格式：请求行（request line）、请求头部（header）、空行和请求数据4个部分组成，下图给出了请求报文的一般格式。



服务器响应消息

HTTP响应也由三个部分组成，分别是：状态行、消息报头、响应正文。

```
HTTP/1.1 200 OK
Date: Sat, 31 Dec 2005 23:59:59 GMT
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 122

<html>
<head>
<title>Wrox Homepage</title>
</head>
<body>
<!-- body goes here -->
</body>
</html>
```

状态行

消息报头

空行

下面的就是响应正文了

实例

下面实例是一点典型的使用GET来传递数据的实例：

客户端请求：

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
```

服务端响应:

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain
```

输出结果：

```
Hello World! My payload includes a trailing CRLF.
```

HTTP请求方法

根据HTTP标准，HTTP请求可以使用多种请求方法。

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。

HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

HTTP请求头信息

HTTP请求头提供了关于请求，响应或者其他的发送实体的信息。在本章节中我们将具体来介绍HTTP请求头信息。

应答头	说明
Allow	服务器支持哪些请求方法（如GET、POST等）。
Content-Encoding	文档的编码（Encode）方法。只有在解码之后才可以得到Content-Type头指定的内容类型。利用gzip压缩文档能够显著地减少HTML文档的下载时间。Java的GZIPOutputStream可以很方便地进行gzip压缩，但只有Unix上的Netscape和Windows上的IE 4、IE 5才支持它。因此，Servlet应该通过查看Accept-Encoding头（即request.getHeader("Accept-Encoding")）检查浏览器是否支持gzip，为支持gzip的浏览器返回经gzip压缩的HTML页面，为其他浏览器返回普通页面。
Content-Length	表示内容长度。只有当浏览器使用持久HTTP连接时才需要这个数据。如果你想要利用持久连接的优势，可以把输出文档写入 ByteArrayOutputStram，完成后查看其大小，然后把该值放入Content-Length头，最后通过 byteArrayStream.writeTo(response.getOutputStream()发送内容。
Content-Type	表示后面的文档属于什么MIME类型。Servlet默认为text/plain，但通常需要显式地指定为text/html。由于经常要设置Content-Type，因此HttpServletResponse提供了一个专用的方法setContentType。
Date	当前的GMT时间。你可以用setDateHeader来设置这个头以避免转换时间格式的麻烦。
Expires	应该在什么时候认为文档已经过期，从而不再缓存它？
Last-Modified	文档的最后改动时间。客户可以通过If-Modified-Since请求头提供一个日期，该请求将被视为一个条件GET，只有改动时间迟于指定时间的文档才会返回，否则返回一个304（Not Modified）状态。Last-Modified也可用setDateHeader方法来设置。
Location	表示客户应当到哪里去提取文档。Location通常不是直接设置的，而是通过HttpServletResponse的sendRedirect方法，该方法同时设置状态代码为302。
Refresh	表示浏览器应该在多少时间之后刷新文档，以秒计。除了刷新当前文档之外，你还可以通过setHeader("Refresh", "5; URL=http://host/path")让浏览器读取指定的页面。注意这种功能通常是通过设置HTML页面HEAD区的＜META HTTP-EQUIV="Refresh" CONTENT="5;URL=http://host/path"＞实现，这是因为，自动刷新或重定向对于那些不能使用CGI或Servlet的HTML编写者十分重要。但是，对于Servlet来说，直接设置Refresh头更加方便。注意Refresh的意义是"N秒之后刷新本页面或访问指定页面"，而不是"每隔N秒刷新本页面或访问指定页面"。因此，连续刷新要求每次都发送一个Refresh头，而发送204状态代码则可以阻止浏览器继续刷新，不管是使用Refresh头还是＜META HTTP-EQUIV="Refresh" ...＞。注意Refresh头不属于HTTP 1.1正

	式规范的一部分，而是一个扩展，但Netscape和IE都支持它。
Server	服务器名字。Servlet一般不设置这个值，而是由Web服务器自己设置。
Set-Cookie	设置和页面关联的Cookie。Servlet不应使用response.setHeader("Set-Cookie", ...)，而是应使用HttpServletResponse提供的专用方法addCookie。参见下文有关Cookie设置的讨论。
WWW-Authenticate	客户应该在Authorization头中提供什么类型的授权信息？在包含401（Unauthorized）状态行的应答中这个头是必需的。例如，response.setHeader("WWW-Authenticate", "BASIC realm=＼"executives＼"")。注意Servlet一般不进行这方面的处理，而是让Web服务器的专门机制来控制受密码保护页面的访问（例如.htaccess）。

HTTP状态码

当浏览者访问一个网页时，浏览者的浏览器会向网页所在服务器发出请求。当浏览器接收并显示网页前，此网页所在的服务器会返回一个包含HTTP状态码的信息头（server header）用以响应浏览器的请求。

HTTP状态码的英文为HTTP Status Code。

下面是常见的HTTP状态码：

- 200 - 请求成功
- 301 - 资源（网页等）被永久转移到其它URL
- 404 - 请求的资源（网页等）不存在
- 500 - 内部服务器错误

HTTP状态码分类

HTTP状态码由三个十进制数字组成，第一个十进制数字定义了状态码的类型，后两个数字没有分类的作用。HTTP状态码共分为5种类型：

分类	分类描述
1**	信息，服务器收到请求，需要请求者继续执行操作
2**	成功，操作被成功接收并处理
3**	重定向，需要进一步的操作以完成请求
4**	客户端错误，请求包含语法错误或无法完成请求
5**	服务器错误，服务器在处理请求的过程中发生了错误

HTTP状态码列表:

状态码	状态码英文名称	中文描述
100	Continue	继续。客户端应继续其请求
101	Switching Protocols	切换协议。服务器根据客户端的请求切换协议。只能切换到更高级的协议，例如，切换到HTTP的新版本协议
200	OK	请求成功。一般用于GET与POST请求
201	Created	已创建。成功请求并创建了新的资源
202	Accepted	已接受。已经接受请求，但未处理完成

203	Non-Authoritative Information	非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本
204	No Content	无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档
205	Reset Content	重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域
206	Partial Content	部分内容。服务器成功处理了部分GET请求
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择
301	Moved Permanently	永久移动。请求的资源已被永久的移动到新URI，返回信息会包括新的URI，浏览器会自动定向到新URI。今后任何新的请求都应使用新的URI代替
302	Found	临时移动。与301类似。但资源只是临时被移动。客户端应继续使用原有URI
303	See Other	查看其它地址。与301类似。使用GET和POST请求查看
304	Not Modified	未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。客户端通常会缓存访问过的资源，通过提供一个头信息指出客户端希望只返回在指定日期之后修改的资源
305	Use Proxy	使用代理。所请求的资源必须通过代理访问
306	Unused	已经被废弃的HTTP状态码
307	Temporary Redirect	临时重定向。与302类似。使用GET请求重定向
400	Bad Request	客户端请求的语法错误，服务器无法理解
401	Unauthorized	请求要求用户的身份认证
402	Payment Required	保留，将来使用
403	Forbidden	服务器理解请求客户端的请求，但是拒绝执行此请求
404	Not Found	服务器无法根据客户端的请求找到资源（网页）。通过此代码，网站设计人员可设置"您所请求的资源无法找到"的个性页面
405	Method Not Allowed	客户端请求中的方法被禁止
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权
408	Request	服务器等待客户端发送的请求时间过长，超时

408	Time-out	服务器等待客户端发送的请求时间过长，超时
409	Conflict	服务器完成客户端的PUT请求是可能返回此代码，服务器处理请求时发生了冲突
410	Gone	客户端请求的资源已经不存在。410不同于404，如果资源以前有现在被永久删除了可使用410代码，网站设计人员可通过301代码指定资源的新位置
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息
412	Precondition Failed	客户端请求信息的先决条件错误
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式
416	Requested range not satisfiable	客户端请求的范围无效
417	Expectation Failed	服务器无法满足Expect的请求头信息
500	Internal Server Error	服务器内部错误，无法完成请求
501	Not Implemented	服务器不支持请求的功能，无法完成请求
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求
503	Service Unavailable	由于超载或系统维护，服务器暂时的无法处理客户端的请求。延时的长度可包含在服务器的Retry-After头信息中
504	Gateway Time-out	充当网关或代理的服务器，未及时从远端服务器获取请求
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理

HTTP content-type

Content-Type，内容类型，一般是指网页中存在的Content-Type，用于定义网络文件的类型和网页的编码，决定浏览器将以什么形式、什么编码读取这个文件，这就是经常看到一些Asp网页点击的结果却是下载到的一个文件或一张图片的原因。

HTTP content-type 对照表

文件扩展名	Content-Type(Mime-Type)	文件扩展名	Content-Type(Mime-Type)
.*（二进制流，不知道下载文件类型）	application/octet-stream	.tif	image/tiff
.001	application/x-001	.301	application/x-301
.323	text/h323	.906	application/x-906
.907	drawing/907	.a11	application/x-a11
.acp	audio/x-mei-aac	.ai	application/postscript
.aif	audio/aiff	.aifc	audio/aiff
.aiff	audio/aiff	.anv	application/x-anv
.asa	text/asa	.asf	video/x-ms-asf
.asp	text/asp	.asx	video/x-ms-asf
.au	audio/basic	.avi	video/avi
.awf	application/vnd.adobe.workflow	.biz	text/xml
.bmp	application/x-bmp	.bot	application/x-bot
.c4t	application/x-c4t	.c90	application/x-c90
.cal	application/x-cals	.cat	application/vnd.ms-pki.seccat
.cdf	application/x-netcdf	.cdr	application/x-cdr
.cel	application/x-cel	.cer	application/x-x509-ca-cert
.cg4	application/x-g4	.cgm	application/x-cgm
.cit	application/x-cit	.class	java/*
.cml	text/xml	.cmp	application/x-cmp

.cmx	application/x-cmx	.cot	application/x-cot
.crl	application/pkix-crl	.crt	application/x-x509-ca-cert
.csi	application/x-csi	.css	text/css
.cut	application/x-cut	.dbf	application/x-dbf
.dbm	application/x-dbm	.dbx	application/x-dbx
.dcd	text/xml	.dcx	application/x-dcx
.der	application/x-x509-ca-cert	.dgn	application/x-dgn
.dib	application/x-dib	.dll	application/x-msdownload
.doc	application/msword	.dot	application/msword
.drw	application/x-drw	.dtd	text/xml
.dwf	Model/vnd.dwf	.dwf	application/x-dwf
.dwg	application/x-dwg	.dxb	application/x-dxb
.dxf	application/x-dxf	.edn	application/vnd.adobe.edn
.emf	application/x-emf	.eml	message/rfc822
.ent	text/xml	.epi	application/x-epi
.eps	application/x-ps	.eps	application/postscript
.etd	application/x-ebx	.exe	application/x-msdownload
.fax	image/fax	.fdf	application/vnd.fdf
.fif	application/fractals	.fo	text/xml
.frm	application/x-frm	.g4	application/x-g4
.gbr	application/x-gbr	.	application/x-
.gif	image/gif	.gl2	application/x-gl2
.gp4	application/x-gp4	.hgl	application/x-hgl
.hmr	application/x-hmr	.hpg	application/x-hpgl
.hpl	application/x-hpl	.hqx	application/mac-binhex40
.hrf	application/x-hrf	.hta	application/hta
.htc	text/x-component	.htm	text/html
.html	text/html	.htt	text/webviewhtml
.htx	text/html	.icb	application/x-icb
.ico	image/x-icon	.ico	application/x-ico
.iff	application/x-iff	.ig4	application/x-g4
.igs	application/x-igs	.iii	application/x-iphone

.img	application/x-img	.ins	application/x-internet-signup
.isp	application/x-internet-signup	.IVF	video/x-ivf
.java	java/*	.jfif	image/jpeg
.jpe	image/jpeg	.jpe	application/x-jpe
.jpeg	image/jpeg	.jpg	image/jpeg
.jpg	application/x-jpg	.js	application/x-javascript
.jsp	text/html	.la1	audio/x-liquid-file
.lar	application/x-laplayer-reg	.latex	application/x-latex
.lavs	audio/x-liquid-secure	.lbm	application/x-lbm
.lmsff	audio/x-la-lms	.ls	application/x-javascript
.ltr	application/x-ltr	.m1v	video/x-mpeg
.m2v	video/x-mpeg	.m3u	audio/mpegurl
.m4e	video/mpeg4	.mac	application/x-mac
.man	application/x-troff-man	.math	text/xml
.mdb	application/msaccess	.mdb	application/x-mdb
.mfp	application/x-shockwave-flash	.mht	message/rfc822
.mhtml	message/rfc822	.mi	application/x-mi
.mid	audio/mid	.midi	audio/mid
.mil	application/x-mil	.mml	text/xml
.mnd	audio/x-musicnet-download	.mns	audio/x-musicnet-stream
.mocha	application/x-javascript	.movie	video/x-sgi-movie
.mp1	audio/mp1	.mp2	audio/mp2
.mp2v	video/mpeg	.mp3	audio/mp3
.mp4	video/mpeg4	.mpa	video/x-mpg
.mpd	application/vnd.ms-project	.mpe	video/x-mpeg
.mpeg	video/mpg	.mpg	video/mpg
.mpga	audio/rn-mpeg	.mpp	application/vnd.ms-project
.mps	video/x-mpeg	.mpt	application/vnd.ms-project
.mpv	video/mpg	.mpv2	video/mpeg
.mpw	application/vnd.ms-project	.mpx	application/vnd.ms-project
.mtx	text/xml	.mxp	application/x-mmxp

.net	image/pnetvue	.nrf	application/x-nrf
.nws	message/rfc822	.odc	text/x-ms-odc
.out	application/x-out	.p10	application/pkcs10
.p12	application/x-pkcs12	.p7b	application/x-pkcs7-certificates
.p7c	application/pkcs7-mime	.p7m	application/pkcs7-mime
.p7r	application/x-pkcs7-certreqresp	.p7s	application/pkcs7-signature
.pc5	application/x-pc5	.pci	application/x-pci
.pcl	application/x-pcl	.pcx	application/x-pcx
.pdf	application/pdf	.pdf	application/pdf
.pdx	application/vnd.adobe.pdx	.pfx	application/x-pkcs12
.pgl	application/x-pgl	.pic	application/x-pic
.pko	application/vnd.ms-pki.pko	.pl	application/x-perl
.plg	text/html	.pls	audio/scpls
.plt	application/x-plt	.png	image/png
.png	application/x-png	.pot	application/vnd.ms-powerpoint
.ppa	application/vnd.ms-powerpoint	.ppm	application/x-ppm
.pps	application/vnd.ms-powerpoint	.ppt	application/vnd.ms-powerpoint
.ppt	application/x-ppt	.pr	application/x-pr
.prf	application/pics-rules	.prn	application/x-prn
.prt	application/x-prt	.ps	application/x-ps
.ps	application/postscript	.ptn	application/x-ptn
.pwz	application/vnd.ms-powerpoint	.r3t	text/vnd.rn-realtext3d
.ra	audio/vnd.rn-realaudio	.ram	audio/x-pn-realaudio
.ras	application/x-ras	.rat	application/rat-file
.rdf	text/xml	.rec	application/vnd.rn-recording
.red	application/x-red	.rgb	application/x-rgb
.rjs	application/vnd.rn-realsystem-rjs	.rjt	application/vnd.rn-realsystem-rjt
.rlc	application/x-rlc	.rle	application/x-rle
.rm	application/vnd.rn-realmedia	.rmf	application/vnd.adobe.rmf
.rmi	audio/mid	.rmj	application/vnd.rn-realsystem-rmj

.rmm	audio/x-pn-realaudio	.rmp	application/vnd.rn-rn_music_package
.rms	application/vnd.rn-realmedia-secure	.rmvb	application/vnd.rn-realmedia-vbr
.rmx	application/vnd.rn-realsystem-rmx	.rnx	application/vnd.rn-realplayer
.rp	image/vnd.rn-realpix	.rpm	audio/x-pn-realaudio-plugin
.rsml	application/vnd.rn-rsml	.rt	text/vnd.rn-realtext
.rtf	application/msword	.rtf	application/x-rtf
.rv	video/vnd.rn-realvideo	.sam	application/x-sam
.sat	application/x-sat	.sdp	application/sdp
.sdw	application/x-sdw	.sit	application/x-stuffit
.slb	application/x-slb	.sld	application/x-sld
.slk	drawing/x-slk	.smi	application/smil
.smil	application/smil	.smk	application/x-smk
.snd	audio/basic	.sol	text/plain
.sor	text/plain	.spc	application/x-pkcs7-certificates
.spl	application/futuresplash	.spp	text/xml
.ssm	application/streamingmedia	.sst	application/vnd.ms-pki.certstore
.stl	application/vnd.ms-pki.stl	.stm	text/html
.sty	application/x-sty	.svg	text/xml
.swf	application/x-shockwave-flash	.tdf	application/x-tdf
.tg4	application/x-tg4	.tga	application/x-tga
.tif	image/tiff	.tif	application/x-tif
.tiff	image/tiff	.tld	text/xml
.top	drawing/x-top	.torrent	application/x-bittorrent
.tsd	text/xml	.txt	text/plain
.uin	application/x-icq	.uls	text/iuls
.vcf	text/x-vcard	.vda	application/x-vda
.vdx	application/vnd.visio	.vml	text/xml
.vpg	application/x-vpeg005	.vsd	application/vnd.visio
.vsd	application/x-vsd	.vss	application/vnd.visio

.vst	application/vnd.visio	.vst	application/x-vst
.vsw	application/vnd.visio	.vsx	application/vnd.visio
.vtx	application/vnd.visio	.vxml	text/xml
.wav	audio/wav	.wax	audio/x-ms-wax
.wb1	application/x-wb1	.wb2	application/x-wb2
.wb3	application/x-wb3	.wbmp	image/vnd.wap.wbmp
.wiz	application/msword	.wk3	application/x-wk3
.wk4	application/x-wk4	.wkq	application/x-wkq
.wks	application/x-wks	.wm	video/x-ms-wm
.wma	audio/x-ms-wma	.wmd	application/x-ms-wmd
.wmf	application/x-wmf	.wml	text/vnd.wap.wml
.wmv	video/x-ms-wmv	.wmx	video/x-ms-wmx
.wmz	application/x-ms-wmz	.wp6	application/x-wp6
.wpd	application/x-wpd	.wpg	application/x-wpg
.wpl	application/vnd.ms-wpl	.wq1	application/x-wq1
.wr1	application/x-wr1	.wri	application/x-wri
.wrk	application/x-wrk	.ws	application/x-ws
.ws2	application/x-ws	.wsc	text/scriptlet
.wsdl	text/xml	.wvx	video/x-ms-wvx
.xdp	application/vnd.adobe.xdp	.xdr	text/xml
.xfd	application/vnd.adobe.xfd	.xfdf	application/vnd.adobe.xfdf
.xhtml	text/html	.xls	application/vnd.ms-excel
.xls	application/x-xls	.xlw	application/x-xlw
.xml	text/xml	.xpl	audio/scpls
.xq	text/xml	.xql	text/xml
.xquery	text/xml	.xsd	text/xml
.xsl	text/xml	.xslt	text/xml
.xwd	application/x-xwd	.x_b	application/x-x_b
.sis	application/vnd.symbian.install	.sisx	application/vnd.symbian.install
.x_t	application/x-x_t	.ipa	application/vnd.iphone
.apk	application/vnd.android.package-archive	.xap	application/x-silverlight-app

Web 主机

网站主机 介绍

互联网是如何工作的呢？如何制作属于您自己的网站？

什么是网站主机? 什么是 Internet 服务提供商（ISP）？

什么是万维网（ **World Wide Web**）？

- Web是一个遍布全球的计算机网络。
- 网络中的所有计算机均可彼此相互通信。
- 所有的计算机都使用被称为 HTTP 的通信标准。

WWW 如何工作？

- Web 信息存储于被称为网页的文档中。
- 网页是存储于名为 web 服务器的计算机中的文件。
- 读取网页的计算机可称为 web 客户机。
- web 客户机通过名为 web 浏览器的程序来查看页面。
- 主流的浏览器有 **Internet Explorer** 和 **Firefox**。

浏览器如何读取网页？

- 浏览器可以通过一个请求 (request) 从 web 服务器读取页面。
- 请求是包含页面地址的标准 HTTP 请求。
- 网页地址实例: **http://www.w3cschool.cc/**

浏览器如何显示网页？

- 所有的网页都含有供显示的指令。
- 浏览器通过读取这些指令来显示页面。
- 最常用的显示指令是 HTML 标签。
- HTML 标签格式：**<p>**这是一个段落。**</p>.**

如果你想学习更多关于HTML的知识请访问我们的 HTML 教程.

什么是 **Web** 服务器？

- 您的所有网页的集合被称为网站。
- 要想让别人看到您的页面，就必须对网站进行发布。
- 您必须把网站拷贝到一台 web 服务器，才能完成对网站的发布。
- 如果您的 PC 连入网络的话，您也可以把它当作一台 web 服务器。
- 大多数的情况是使用由 ISP 提供的 web 主机。

什么是 ISP(Internet Service Provider)?

- ISP 指的是 Internet 服务提供商。
- ISP 可提供 Internet 服务。
- 最常见的 Internet 服务是网站主机。
- 网站主机服务可把您的网站存放到一个公共的服务器上。
- 网站主机服务通常包括了域名注册服务。

总结

如果您希望其它人看到您的网站，就必须把网站拷贝到一个公共的服务器。即使您可以使用自己的 PC 来做 web 服务器，最常见的做法还是通过 ISP 来存放网站。

包含在 web 主机解决方案中的还有域名注册和标准的电子邮件服务。

您可以在接下来的章节阅读到更多有关域名注册、电子邮件和其他服务的内容。

网站主机提供商

如果您希望让全世界的人都看到您的网站，就必须把它存储在一个 web 服务器。

使用自己的主机

在自己的服务器上存放网站始终都是一个选项。不过有些问题是需要考虑的：

硬件的费用

要运行一个"真正"的网站，你必须购买一些高性能的服务器硬件。不要指望低价的 PC 可以做这些工作。同时您还需要可以到达您办公室的不间断高速连接，而这种连接是很昂贵的。

软件费用

不要忘记为软件许可计算额外的费用。请记住服务器许可的价格通常会远高于客户机许可。同时还需要注意某些服务器软件的许可会限制并发用户的数量。

劳务费用

不要指望很低的劳务费用。请记住您不得不安装这些软件和硬件，还需要对付漏洞和病毒，并在一个"可能发生任何事情"的环境里保持服务器的不间断运行。

使用互联网服务提供商（**ISP**： **Internet Service Provider**）

从互联网服务提供商（ISP）租用服务器，是一种常见的选择。

从一个 ISP 租用服务器是最常见的做法。这样做的好处有：

连接速度

大多数提供商都拥有极快的 Internet 连接。

强大的硬件

服务提供商通常拥有很多台可供多家公司分享的强大的 web 服务器。它们可提供负载平衡以及必要的备份服务。

安全和稳定性

ISP 是网站主机领域的专家。他们能够提供超过 99% 的正常服务时间、最新的软件漏洞补丁以及最好的病毒防护。

使用 ISP 需要考虑的事情

24小时支持

要确保你选择的 ISP 提供24小时的支持。请不要把自己置于需要等待下一个工作日才能修复紧要问题的境地。如果您不希望花费大量的长途电话费，被叫免费电话也是很重要的。

每日的备份

要确保您选择的 ISP 提供每日例行的安全备份，否则你可能会损失不少有价值的数据。

流量限制

请研究一下提供商的流量限制条款。要确保如果您的网站开始受欢迎以后，您不必为未预料到的高流量付额外的费用。

带宽或内容限制

请研究一下提供商的带宽和内容限制条款。如果您计划发布图片、广播或者声音，要确定您是否有这样的权利。

电子邮件性能

请确保提供商可完全地支持您需要的电子邮件性能。（您可以在后面的章节获得更多有关电子邮件性能的信息）

Front Page 扩展

如果您计划使用 FrontPage 来开发网站的话，请确保您的提供商可完全地支持 FrontPage 服务器扩展。

数据库访问

如果您计划在您的网站使用数据库，请确保您的提供商完全地支持您所需要的数据库访问。（您可以在后面的章节阅读到更多数据库访问的内容）

网站 域名

域名是网站唯一的名称。

主机解决方案中应包括域名注册。

域名应该容易记、容易写。

什么是域名？

域名是网站的唯一名称，比如 **w3cschool.cc**。

域名是需要注册的。当域名被注册后，就会被添加到大的域名注册商那里，连同与您的网站有关的信息 - 包括被保存在 DNS 服务器的 IP 信息。

DNS 指的是域名系统 (Domain Name System)。DNS 服务器负责向 internet 上的其他计算机通知有关你的域名和地址的信息。

注册域名

可以通过域名注册公司来注册域名。

这些公司都提供了查询可用域名的接口，并提供可同时注册的域名后缀。

选择域名

对于任何个人或组织来说，选择域名都是很重要的步骤。

在域名被大量注册的同时，新的域名后缀和创造性的思路仍然可提供数千种很棒的选择。

当选择一个域名时，考虑好域名的用途是很重要的，这会使人们更容易找到你的网站。

优秀的域名具有以下特征：

简短 - 人们都不喜欢打字！域名越短，就更容易被访问，用户发生输入错误的可能性也就越小。

有意义 - 没有意义的短域名不见得好。34i4nh69.com 仅有8个字符长，但是既不易输入也不易记忆。所以，还是请您选择与您的网站相关且容易理解的域名吧。

清晰- 在选择域名时，清晰也很重要。避免选择那些很难拼写或发音的域名。另外还要注意，您所选的域名听起来是不是顺耳，通过电话能否很快地交流。

曝光率 - 就像高档的房地产项目可获得极高的曝光率一样，短而容易记的域名也是一种资产。除了访问者使用您的域名以外，你还应该考虑到搜索引擎。搜索引擎是通过人们在线搜索项目的相关度来对您的网站进行索引和评级的。为了最大程度地使您的网站曝光，可以在域名中包含相关的搜索项。当然，前提仍然是你选择的域名应该是简短的、清晰的且有意义的。

子域名

大多数人们都没有意识到，但确实是每天都在使用着子域名。最常见的 "www" 其实就是典型的子域名。子域名可以在 DNS

服务器上创建，并且不需要通过域名注册机构来进行注册。当然，在创建子域名之前，还是需要首先注册原始域名的。在 internet 上可以见到很多子域名的例子：比如 store.apple.com 和 support.microsoft.com。

可请求您的网站主机提供商来创建子域名，也可以通过管理您的 DNS 服务器来创建。

虚假域名 - 目录列表

某些提供商可能会为您提供一个位于其域名之下的一个唯一的名称：www.theircompany.com/yourcompany/

这并不是一个真实的域名，而是一个目录 - 应当尽量避免这样的情况。

这样的 URL 是不值得要的，特别是对公司来说。所以还是避免使用它吧，如果您能够支付域名的费用的话。这种 URL 的典型运用是通过 ISP 用于个人网站或免费网站，其实就是分享一个独立域名的方式，可为用户提供属于自己的地址。

域名注册机构之间的完全竞争已经极大程度地降低了域名的注册费用，所有这种域名分享的方式越来越少见了，因为人们仅仅需要 15 美元就可以注册属于自己的域名了。

过期的域名

域名注册的另一个来源是过期的域名。当您注册了一个域名后，假设没有法律或商标方面的争议，那么只要您付清费用（您可以提前支付 10 年的费用），就可以自由地使用任意长的时间。某些人是利用域名进行投机的，希望以后可以卖掉它们，而另一些人本来计划使用某个域名却没有这么做。结果是之前已被注册的域名会定期地成为可注册的状态。您可以通过 http://www.dotdnr.com 这个网站查询最近的过期域名，如果希望注册的话，需要付的费用和注册新域名是相同的。

使用您的域名

在您已经选择 - 并注册了 - 属于自己的域名后，一定要把它用在您所有的网页上，还有所有的信件上，比如电子邮件和传统的信件。

让其他人知晓您的域名是很重要的，把您的域名告诉您的合作伙伴和客户吧。

网站主机 性能

首先要确定您所需要的磁盘空间和流量。

磁盘空间大小？

小型或中型的网站至少需要 10MB 到 100MB 的磁盘空间。

如果只考虑 HTML 页面的话，它们的平均尺寸是很小的。也许甚至不到 1KB。但是如果看一下在页面中使用的图形的尺寸，您会发现大多数图片的尺寸要比页面本身大得多。

加上图片和其他一些占用空间的元素，每张页面会占用 5KB 到 50KB 的服务器空间。

如果您计划使用大量的图像和图形元素（不涉及音频文件和视频文件），那么可能需要更多的磁盘空间。

在您挑选提供商之前，请首先明确您需要的磁盘空间。

月流量

小型或中型的网站每月至少需要 1GB 到 5GB 的数据传输量。

可以这样进行计算：把平均的页面尺寸乘以每月预期的页面浏览量。假设您的平均页面尺寸是 30KB，预期的页面浏览量是 50,000 张页面，那么您需要 0.03MB x 50,000 = 1.5GB 。

更大的商业站点每月通常会消耗掉不少于 100GB 的流量。

在与主机提供商签合同之前，需要搞清楚下面的事项：

- 月流量限制是多少？
- 如果超过限制，网站会被关闭吗？
- 如果超过限制，需要付额外的费用吗？
- 主机容易进行升级吗？

连接速度

访问者通常会使用调制解调器来访问您的网站，但是主机提供商则拥有极高的连接速度。

在 Internet 的早期，T1 连接被认为是非常快的连接。而今天的连接速度则要快得多。

1 字节等于 8 比特（这是用于传输一个字符的比特数），低速调制解调器能够传输大概 14 000 到 56 000 比特每秒（14 至 56 千比特每秒），即每秒传输 2000 至 7000 个字符，或大约 1 到 5 页文本。

一个千比特 (Kb) 是 1024 比特。一个兆比特 (Mb) 是 1024 千比特。一个 gigabit (Gb) 是 1024 兆比特。

这是目前在 Internet 上被使用到的连接速度：

名称	连接	每秒的速度
Modem	Analog	14.4-56Kb
D0	Digital(ISDN)	64Kb
T1	Digital	1.55Mb
T3	Digital	43Mb
OC-1	Optical Carrier	52Mb
OC-2	Optical Carrier	156Mb
OC-12	Optical Carrier	622Mb
OC-24	Optical Carrier	1.244Gb
OC-48	Optical Carrier	2.488Gb

在签合同之前，可以试试提供商服务器上的其他网站，与其他的客户交流一下也是不错的做法。

主机 电子邮件访问

主机服务应该包括一定的电子邮件帐号和电子邮件服务器。

E-mail 账号

主机解决方案应该有能力为公司中的每个人提供一个电子邮件帐号。

E-mail 地址格式如下所示:

john@mycompany.com

john.doe@mycompany.com

jdoe@mycompany.com

POP E-mail

POP 指的是邮局协议。POP 是一种用于发送和接收电子邮件的标准客户机/服务器协议。

电子邮件会被接收并保存到您的 internet 服务器上，直到您通过某个客户段邮件程序（比如 Outlook 和 Foxmail）来收取信件。

IMAP Email

IMAP 指的是 Internet 消息访问协议。IMAP 是另外一种用于发送和接收电子邮件的标准协议。

IMAP 在 POP 的基础上提供了一定的改进，即存储在 IMAP 服务器上的电子邮件可以由多台计算机处理，而无需在计算机间来回传输消息。而 POP 被设计为支持在一台单独的计算机上进行的邮件访问。

基于 web 的电子邮件

基于 web 的电子邮件使我们通过 web 浏览器就可以访问电子邮件。您通过 web 登陆到电子邮件帐户以后就可以发送和接收电子邮件了。能够从世界上的任何地方访问电子邮件是件很吸引人的事情。

基于 web 的电子邮件的典型例子有：Gmail 和 Hotmail 。

邮件转发

电子邮件转发使我们能够拥有多个邮件名。

通过电子邮件转发功能，可以为别的邮件帐户设置别名：

例如：

可以把发往 postmaster@mycompany.com 的邮件转发到 peter@mycompany.com

把发往 sales@mycompany.com 的邮件转发到 mary@mycompany.com

邮件列表

某些服务器上会提供邮件列表功能。如果您希望向大量的用户发送邮件的话，这是一项有价值的功能。

网站主机 技术

本节介绍一些最常用的的主机技术。

Windows 主机

Windows 主机是运作在 Windows 操作系统上的主机服务。

如果您使用ASP作为服务器脚本，或者计划使用微软的 Access 或 SQL Server 数据库的话，就应该选择 Windows 平台的主机。另外，如果您计划使用 Microsoft Front Page 来开发网站的话，Windows 主机也是最佳的选择。

Unix 主机

Unix 主机是运作在 Unix 操作系统上的主机服务。

Unix 是首个（或最原始的）web 服务器操作系统，并以可靠性和稳定性而闻名。而且价格也通常低于 Windows 。

Linux 主机

Linux 主机是运作在 Linux 操作系统上的主机服务。

CGI

网页可作为 CGI 脚本来执行。CGI 脚本可在服务器上执行，来生成动态的交互性页面。

大多数的 ISP 都会提供对 CGI 的某种程度的支持。并且许多都提供了使用 CGI 编写的预先安装的可运行的留言簿、页面计数器以及聊天/论坛解决方案。

CGI 最常使用在 Unix 或 Linux 服务器。

ASP - Active Server Pages

ASP 是由微软公司研发的服务器端脚本技术。

通过把脚本代码放到 HTML 页面内，您可以使用 ASP 来创建动态的网页。在页面返回浏览器之前，代码会首先被服务器执行。而且 Visual Basic 和 JavaScript 都可使用。

ASP 是 Windows 95,98, 2000 以及 XP 中的标准组件。可在所有运行 Windows 的计算机上激活 ASP 。

许多的主机提供商都提供 ASP 支持，ASP 技术在中国已经很流行了。

如果您需要学习更多有关 ASP 的知识，请访问我们的 ASP 教程。

PHP

类似 ASP，PHP 也是一门服务器端脚本语言，通过把脚本代码放到 HTML 页面内，您可以使用 PHP 来创建动态的网页。在页面返回浏览器之前，代码会首先被服务器执行。

PHP非常适合用于Web开发，HTML代码中可以嵌入PHP代码。

PHP 语法类似于 Perl 和 C。

在各种操作系统上，PHP往往是与Apache（Web服务器）一起使用。它也支持ISAPI及微软的Windows IIS。

PHP支持很多数据库，如MySQL和Informix，Oracle，SYBASE，Solid，PostgreSQL，Generic ODBC等。

如果你想学习更多关于PHP的知识，请访问我们的 PHP 教程。

JSP

JSP 是一种由 SUN 开发的类似 ASP 的服务器端技术。

使用 JSP，您可以通过把 Java 代码放入 HTML 页面来创建动态页面。在页面返回浏览器之前，代码同样会首先被服务器执行。

由于 JSP 使用 Java，此技术不会受限于任何的服务器平台。

Cold Fusion

Cold Fusion 是另一门用来创建动态网页的服务器端脚本语言。

Cold Fusion 是由 Macromedia 开发的。

Chili!Soft ASP

微软的 ASP 技术只能运行在 Windows 平台。

不过，Chili!Soft ASP 则是一种使得 ASP 可运行在 UNIX 和其他平台的软件产品。

Microsoft Expression Web

Expression Web 由微软开发的网站设计工具。

Expression Web 可以让初级开发者轻松开发网站。

如果你打算使用 Expression Web, 你需要查看 Windows 主机解决方案。

Adobe Dreamweaver

Dreamweaver是一个由Adobe Systems所拥有的网站设计工具。

在用户不具备深入的web开发知识的情况下，就可以使用Dreamweaver开发出一个网站。

Dreamweaver支持CSS，JavaScript，ASP.NET，ColdFusion，JavaServer Pages，和PHP等Web技术开发。

Dreamweaver可在Mac和Windows两种操作系统上运行。

安全服务器

一个安全的服务器可以对传输的数据进行加密。

如果你打算做网上的信用卡交易，或其他类型的Web通信，需要加以保护防止未经授权的访问，您的ISP必须提供一个安全的服务器。

网站 数据库

MS SQL Server 或者 Oracle 用于高流量的数据库驱动型网站。

MySQL用于低成本的数据库访问。

MS Access 用于低流量的网站。

网站数据库

如果您的网站需要经由 web 来更新大量的信息，那么您就需要数据库来存储信息。

可用于网站主机的数据库系统有很多种类型。最常见的是 MS Access、MySQL、SQL Server 以及 Oracle。

使用 SQL 语言

SQL 是一门用于访问数据库的语言。

如果您希望您的网站有能力在数据库存储或检索数据，那么您的 web 服务器就需要使用 SQL 语言对数据库系统进行访问的权限。

如果您希望学习更多有关 SQL 的知识，请访问我们的 SQL 教程 。

SQL Server

微软的 SQL Server 是用于高流量的数据库驱动网站的最流行的数据库软件之一。

SQL Server 是非常强大、健壮且特性丰富的 SQL 数据库系统。

Oracle

Oracle 同样是非常流行的用于高流量数据库驱动网站的数据库软件。

Oracle 同样是非常强大、健壮且特性丰富的 SQL 数据库系统。

MySQL

MySQL 同意是一个流行的网站数据库软件。

MySQL是一个非常强大的和完整的SQL数据库系统。

MySQL 是昂贵的 Microsoft 和 Oracle 解决方案的廉价替代品。

Access

如果网站需要的是一套简易的数据库解决方案，微软的 Access 应该是很受欢迎的选项。

Access 不适合高流量的网站，并且也没有 Oracle 或 SQL Server 那么强大。

网站主机 类型

网站主机的类型有：免费主机、虚拟（分享的）主机或独享主机。

一些服务提供商会提供免费的网站主机。

免费主机适合小型的低流量站点，比如个人网站。但是不推荐高流量或商业网站使用免费的主机，因为常会有技术上的限制，能够选择的选项也很少。

通常，您无法在免费主机上使用自己的域名。而不得不使用由主机提供的地址，类似这样：http://www.freesite/users/~yoursite.htm。这样的 URL 难写、难记，也很不专业。

优点	缺点
费用低。免费的	没有域名。
适合家庭、业务爱好或个人站点。	极少的、有限制的或根本没有软件选项。
通常会有免费的电子邮件。	有限的安全选项。

有限的或没有数据库支持 | 有限的技术支持 |

分享型的主机（虚拟主机）

虚拟主机是最常见的、也是最合算的。

使用虚拟主机的话，您的网站与其余的也许是 100 个网站会被寄存在同一个高性能的服务器上。在一个虚拟主机上，每一个网站都可以使用属于自己的域名。

虚拟主机通常会提供多种软件解决方案，比如电子邮件、数据库、许多不同的编辑选项。技术支持往往也不错。

优点	缺点
低成本。与其他用户分担费用。	由于许多站点在一台服务器上而降低的安全性。
适合小型的商业和中等的流量。	在流量上有限制。
多种软件选项。	有限制的数据库支持。
自己的域名。	有限制的软件支持。
良好的服务支持。	

专享主机

您可以通过专享的主机服务把网站存放到一个专用的服务器上。

专享主机是最昂贵的主机类型。这种解决方案适合大型的高流量网站，以及使用特殊软件的网站。

专享主机的性能很强，也很安全，软件方案也几乎没有限制。

优点	缺点
适合大型商业网站。	昂贵。
适合高流量的昂站。	需要较高的技术。
可使用多个域名。	
强大的电子邮件解决方案。	
强大的数据库支持。	
强有力的（没有限制的）软件支持。	

托管主机

这种解决方案是把您自己的服务器放到服务提供商那里。

这很类似于在您自己的办公室来运行您的服务器，不同的是服务器被安放到一个专门为它设计的场所。

通常提供商都拥有专用的机房资源，比如防火防止故意破坏的高安全性、不间断电源、专用的 Internet 连接等等。

优点	缺点
高带宽。	昂贵
高正常运行时间。	需要更高的技术。
高安全性。	难以设置及修复漏洞。
无限制的软件选项。	

您的清单

在您选择网站主机之前，请首先确定下面的事项：

- 符合您目前需求的主机类型
- 主机类型的性价比
- 能否升级到更好的服务器
- 如果需要的话，能否升级到专享服务器

在您与主机提供商签合同之前，可以访问一下其服务器上的其它网站，感觉一下他们的网速。同时把其它的网站与您的网站做个对比，看一下是否您也有同样的需求。与其他的用户沟通一下也是不错的做法。

电子商务网站主机

如果您的公司正在销售某种产品或服务，那么电子商务也许是做生意的一种好办法。

电子商务

电子商务就是在 Internet 上销售产品或服务。

电子商务系统

对于小公司来说，自己建立一套电子商务系统不是一个理想的做法。构建一套电子商务系统是一个复杂的过程，其中可能会出现很多潜在的错误。

您可以购买一套现成的系统，在自己的服务器上运行。目前市场上有很多系统，其中大部分都可满足您的基本需要，比如订单管理和处理。但是还要指出的是，如果您对网站的主机不是很熟悉，最好不要马上就开始运行电子商务网站。

最好的解决方案，在我们的看来，就是选择一家能够提供电子商务解决方案的主机提供商。

电子商务主机提供商

电子商务可涵盖大范围的产品。而通过不同的提供商，您能够选择从极简单到极复杂的解决方案。

大多数提供商提供简单的价格不高的解决方案，您可借此来运行您自己的"虚拟商店"。

您的检验表

- 如何处理客户信息？
- 如何处理产品目录？
- 如何处理订单？
- 如何处理库存？
- 如何处理延期交货？
- 如何处理货运？
- 如何处理帐目？
- 如何处理支付？
- 如何处理国外货币？
- 如何处理信用卡？

- 如何处理税务？
- 如何解决安全问题？
- 如何解决完整性问题（加密技术）？

还要检查一下这些相当费时的任务是否能够自动完成。比如自动开单据、发票处理、记帐以及报告生成。

税收问题

对网上商店来说，税是一个复杂的问题, 特别是增值税。

如果你上线卖东西，你很可能会有出口业务。

如果是出口货物，你的客户在提货时往往需要交付增值税。

此外，你网店的税收往往取决于你的销售财务报表。

开网店前，请务必咨询税务顾问。

图片服务器

如果您的网站存在大量的图片读写操作，我们建议您使用图片服务器。

通过使用独立的图片服务器，您可以提高网站性能，改善用户体验，并降低运营成本。

什么是图片服务器

图片服务器是专门为图片读写操作优化的独立服务器。

运行网站的服务器称为 Web 服务器。

通过 Web 服务器，用户可以访问静态网页、Web 应用程序、数据库，或者上传下载图片以及其他多媒体内容。

但是，如果网站访问量不断增加，访问速度日趋缓慢，那么就应该考虑将部分功能从 Web 服务器中分离出来。

通常，如果网站存在大量图片读写操作，那么应该首先把图片服务分离出来，也就是建立独立的图片服务器。

图片服务器的优势

总得来说，部署图片服务器有以下几点好处：

- 分担 Web 服务器的 I/O 负载 - 将耗费资源的图片服务分离出来，提高服务器的性能和稳定性
- 能够专门对图片服务器进行优化 - 为图片服务设置有针对性的缓存方案，减少带宽成本，提高访问速度
- 提高网站的可扩展性 - 通过增加图片服务器，提高图片吞吐能力

建立图片服务器的注意事项

- 选择适合图片存储的物理介质和文件系统
- 使用物理上独立的服务器
- 如果拥有多台图片服务器，要考虑服务器之间的图片同步问题
- 使用独立域名
- 制定合理的缓存策略
- 使用图片处理模块对用户上传的图片进行再加工

图片云存储服务

如果您正在运营中小型网站，或者是互联网创业团队，那么使用第三方的图片云存储服务可以获得以下好处：

1. 减少图片服务器的部署时间

2. 降低开发成本

3. 节约宝贵的资金

案例：又拍云存储

又拍云是通用的大规模存储服务，主要为用户提供静态文件存储以及 CDN 加速的服务。

又拍云在静态文件存储方面有多年的技术经验，一直专注于静态文件存储处理领域。

又拍云存储在全国各地有 26 个 CDN 节点， 300 多台服务器以及电信、联通、移动和教育网四线带宽，能够让用户以极低的价格获得可靠、安全和快速的基础存储服务。

了解又拍云服务的更多信息：

又拍云存储的多项优势

又拍云存储的各种解决方案

Web TCP/IP

TCP/IP 介绍

TCP/IP 是用于因特网 (Internet) 的通信协议。

计算机通信协议（Computer Communication Protocol）

计算机通信协议是对那些计算机必须遵守以便彼此通信的的规则的描述。

什么是 TCP/IP？

TCP/IP 是供已连接因特网的计算机进行通信的通信协议。

TCP/IP 指传输控制协议/网际协议（_T_ransmission _C_ontrol _P_rotocol / _I_nternet _P_rotocol）。

TCP/IP 定义了电子设备（比如计算机）如何连入因特网，以及数据如何在它们之间传输的标准。

在 TCP/IP 内部

在 TCP/IP 中包含一系列用于处理数据通信的协议：

- TCP (传输控制协议) - 应用程序之间通信
- UDP (用户数据包协议) - 应用程序之间的简单通信
- IP (网际协议) - 计算机之间的通信
- ICMP (因特网消息控制协议) - 针对错误和状态
- DHCP (动态主机配置协议) - 针对动态寻址

TCP 使用固定的连接

TCP 用于应用程序之间的通信。

当应用程序希望通过 TCP 与另一个应用程序通信时，它会发送一个通信请求。这个请求必须被送到一个确切的地址。在双方"握手"之后，TCP 将在两个应用程序之间建立一个全双工 (full-duplex) 的通信。

这个全双工的通信将占用两个计算机之间的通信线路，直到它被一方或双方关闭为止。

UDP 和 TCP 很相似，但是更简单，同时可靠性低于 TCP。

IP 是无连接的

IP 用于计算机之间的通信。

IP 是无连接的通信协议。它不会占用两个正在通信的计算机之间的通信线路。这样，IP 就降低了对网络线路的需求。每条线可以同时满足许多不同的计算机之间的通信需要。

通过 IP，消息（或者其他数据）被分割为小的独立的包，并通过因特网在计算机之间传送。

IP 负责将每个包路由至它的目的地。

IP 路由器

当一个 IP 包从一台计算机被发送，它会到达一个 IP 路由器。

IP 路由器负责将这个包路由至它的目的地，直接地或者通过其他的路由器。

在一个相同的通信中，一个包所经由的路径可能会和其他的包不同。而路由器负责根据通信量、网络中的错误或者其他参数来进行正确地寻址。

TCP/IP

TCP/IP 意味着 TCP 和 IP 在一起协同工作。

TCP 负责应用软件（比如您的浏览器）和网络软件之间的通信。

IP 负责计算机之间的通信。

TCP 负责将数据分割并装入 IP 包，然后在它们到达的时候重新组合它们。

IP 负责将包发送至接受者。

TCP/IP 寻址

TCP/IP 使用 32 个比特或者 4 组 0 到 255 之间的数字来为计算机编址。

IP地址

每个计算机必须有一个 IP 地址才能够连入因特网。

每个 IP 包必须有一个地址才能够发送到另一台计算机。

在本教程下一节，您会学习到更多关于 IP 地址和 IP 名称的知识。

IP 地址包含 4 组数字：

TCP/IP 使用 4 组数字来为计算机编址。每个计算机必须有一个唯一的 4 组数字的地址。

每组数字必须在 0 到 255 之间，并由点号隔开，比如：192.168.1.60。

32 比特 = 4 字节

TCP/IP 使用 32 个比特来编址。一个计算机字节是 8 比特。所以 TCP/IP 使用了 4 个字节。

一个计算机字节可以包含 256 个不同的值：

00000000、00000001、00000010、00000011、00000100、00000101、00000110、00000111、00001000 .... 直到 11111111。

现在，您应该知道了为什么 TCP/IP 地址是介于 0 到 255 之间的 4 组数字。

IP V6

IPv6 是 "Internet Protocol Version 6" 的缩写，也被称作下一代互联网协议，它是由 IETF 小组（Internet 工程任务组Internet Engineering Task Force）设计的用来替代现行的 IPv4（现行的）协议的一种新的 IP 协议。

我们知道，Internet 的主机都有一个唯一的 IP 地址，IP 地址用一个 32 位二进制的数表示一个主机号码，但 32 位地址资源有限，已经不能满足用户的需求了，因此 Internet 研究组织发布新的主机标识方法，即 IPv6。

在 RFC1884 中（RFC 是 Request for Comments document 的缩写。RFC 实际上就是 Internet 有关服务的一些标准），规定的标准语法建议把 IPv6 地址的 128 位（16 个字节）写成 8 个 16 位的无符号整数，每个整数用 4 个十六进制位表示，这些数之间用冒号（:）分开，例如：3ffe:3201:1401:1280:c8ff:fe4d::db39。

域名

12 个阿拉伯数字很难记忆。使用一个名称更容易。

用于 TCP/IP 地址的名字被称为域名。w3cschool.cc 就是一个域名。

当你键入一个像 http://www.w3cschool.cc 这样的域名，域名会被一种 DNS 程序翻译为数字。

在全世界，数量庞大的 DNS 服务器被连入因特网。DNS 服务器负责将域名翻译为 TCP/IP 地址，同时负责使用新的域名信息更新彼此的系统。

当一个新的域名连同其 TCP/IP 地址一起注册后，全世界的 DNS 服务器都会对此信息进行更新。

TCP/IP 协议

TCP/IP 是不同的通信协议的大集合。

协议族

TCP/IP 是基于 TCP 和 IP 这两个最初的协议之上的不同的通信协议的大集合。

TCP - 传输控制协议

TCP 用于从应用程序到网络的数据传输控制。

TCP 负责在数据传送之前将它们分割为 IP 包，然后在它们到达的时候将它们重组。

IP - 网际协议（Internet Protocol）

IP 负责计算机之间的通信。

IP 负责在因特网上发送和接收数据包。

HTTP - 超文本传输协议(Hyper Text Transfer Protocol)

HTTP 负责 web 服务器与 web 浏览器之间的通信。

HTTP 用于从 web 客户端（浏览器）向 web 服务器发送请求，并从 web 服务器向 web 客户端返回内容（网页）。

HTTPS - 安全的 HTTP（Secure HTTP）

HTTPS 负责在 web 服务器和 web 浏览器之间的安全通信。

作为有代表性的应用，HTTPS 会用于处理信用卡交易和其他的敏感数据。

SSL - 安全套接字层（Secure Sockets Layer）

SSL 协议用于为安全数据传输加密数据。

SMTP - 简易邮件传输协议（Simple Mail Transfer Protocol）

SMTP 用于电子邮件的传输。

MIME - 多用途因特网邮件扩展（Multi-purpose Internet Mail Extensions）

MIME 协议使 SMTP 有能力通过 TCP/IP 网络传输多媒体文件，包括声音、视频和二进制数据。

IMAP - 因特网消息访问协议（Internet Message Access Protocol）

IMAP 用于存储和取回电子邮件。

POP - 邮局协议（Post Office Protocol）

POP 用于从电子邮件服务器向个人电脑下载电子邮件。

FTP - 文件传输协议（File Transfer Protocol）

FTP 负责计算机之间的文件传输。

NTP - 网络时间协议（Network Time Protocol）

NTP 用于在计算机之间同步时间（钟）。

DHCP - 动态主机配置协议（Dynamic Host Configuration Protocol）

DHCP 用于向网络中的计算机分配动态 IP 地址。

SNMP - 简单网络管理协议（Simple Network Management Protocol）

SNMP 用于计算机网络的管理。

LDAP - 轻量级的目录访问协议（Lightweight Directory Access Protocol）

LDAP 用于从因特网搜集关于用户和电子邮件地址的信息。

ICMP - 因特网消息控制协议（Internet Control Message Protocol）

ICMP 负责网络中的错误处理。

ARP - 地址解析协议（Address Resolution Protocol）

ARP - 用于通过 IP 来查找基于 IP 地址的计算机网卡的硬件地址。

RARP - 反向地址转换协议（Reverse Address Resolution Protocol）

RARP 用于通过 IP 查找基于硬件地址的计算机网卡的 IP 地址。

BOOTP - 自举协议（Boot Protocol）

BOOTP 用于从网络启动计算机。

PPTP - 点对点隧道协议（Point to Point Tunneling Protocol）

PPTP 用于私人网络之间的连接（隧道）。

TCP/IP 邮件

电子邮件是 TCP/IP 最重要的应用之一。

您不会用到...

当您写邮件时，您不会用到 TCP/IP。

当您写邮件时，您用到的是电子邮件程序，例如莲花软件的 Notes，微软公司出品的 Outlook，或者 Netscape Communicator 等等。

邮件程序会用到...

您的电子邮件程序使用不同的 TCP/IP 协议：

- 使用 SMTP 来发送邮件
- 使用 POP 从邮件服务器下载邮件
- 使用 IMAP 连接到邮件服务器

SMTP - 简单邮件传输协议

SMTP 协议用于传输电子邮件。SMTP 负责把邮件发送到另一台计算机。

通常情况下，邮件会被送到一台邮件服务器（SMTP 服务器），然后被送到另一台（或几台）服务器，然后最终被送到它的目的地。

SMTP 也可以传送纯文本，但是无法传输诸如图片、声音或者电影之类的二进制数据。

SMTP 使用 MIME 协议通过 TCP/IP 网络来发送二进制数据。MIME 协议会将二进制数据转换为纯文本。

POP - 邮局协议

POP 协议被邮件程序用来取回邮件服务器上面的邮件。

假如您的邮件程序使用 POP，那么一旦它连接上邮件服务器，您的所有的邮件都会被下载到邮件程序中（或者称之为邮件客户端）。

IMAP - 因特网消息访问协议

与 POP 类似，IMAP 协议同样被邮件程序使用。

IMAP 协议与 POP 协议之间的主要差异是：如果 IMAP 连上了邮件服务器，它不会自动地将邮件下载到邮件程序之中。

IMAP 使您有能力在下载邮件之前先通过邮件服务器端查看他们。通过 IMAP，您可以选择下载这些邮件或者仅仅是删除它们。比方说您需要从不同的位置访问邮件服务器，但是仅仅希望回到办公室的时候再下载邮件，IMAP 在这种情况下会很有用。

Web W3C

W3C 简介

什么是 W3C？

- W3C 指万维网联盟（*World Wide Web Consortium*）
- W3C 创建于*1994*年*10*月
- W3C 由 *Tim Berners-Lee* 创建
- W3C 是一个会员组织
- W3C 的工作是对 *web* 进行标准化
- W3C 创建并维护 *WWW* 标准
- W3C 标准被称为 *W3C* 推荐（*W3C* 规范）

W3C 是如何创建的？

万维网（World Wide Web）是作为欧洲核子研究组织的一个项目发展起来的，这那里 Tim Berners-Lee 开发出万维网的雏形。

Tim Berners-Lee - 万维网的发明人 - 目前是万维网联盟的主任。

W3C 在 1994 年被创建的目的是，为了完成麻省理工学院（MIT）与欧洲粒子物理研究所（CERN）之间的协同工作，并得到了美国国防部高级研究计划局（DARPA）和欧洲委员会（European Commission）的支持。

标准化 web

W3C 致力于实现所有的用户都能够对 web 加以利用（不论其文化教育背景、能力、财力以及其身体残疾）。

W3C 同时与其他标准化组织协同工作，比如 Internet 工程工作小组（Internet Engineering Task Force）、无线应用协议（WAP）以及 Unicode 联盟（Unicode Consortium）。

W3C 由美国麻省理工学院计算机科学和人工智能实验室 (MIT CSAIL)，总部位于法国的欧洲信息数学研究联盟(ERCIM) 和日本的庆应大学（Keio University）联合运作，并且在世界范围内拥有分支办事处。

W3C 成员

正因为 Web 是如此的重要（不论在其影响范围还是在投资方面），以至于不应由任何一家单独的组织来对它的未来进行控制，因此 W3C 扮演者一个会员组织的角色：

一些知名的会员包括：

- IBM
- Microsoft
- America Online
- Apple
- Adobe
- Macromedia
- Sun Microsystems

W3C 的会员包括了：软件开发商、内容提供商、企业用户、通信公司、研究机构、研究实验室、标准化团体以及政府。

W3C Recommendations

W3C 最重要的工作是发展 Web 规范（称为推荐，Recommendations），这些规范描述了 Web 的通信协议（比如 HTML 和 XHTML）和其他的构建模块。

每项 W3C 推荐的发展是通过由会员和受邀专家组成的工作组来完成的。工作组的经费来自公司和其他组织，并会创建一个工作草案，最后是一份提议推荐。一般来说，为了获得正式的批准，推荐都会被提交给 W3C 会员和主任。

下一节，我们会为您解释规范的批准过程。

W3C 程序

W3C 的标准化程序分为 7 个不同的步骤。

W3C 规范的批准步骤

在 W3C 发布某个新标准的过程中，规范是通过下面的严格程序由一个简单的理念逐步确立为推荐标准的：

- W3C 收到一份提交
- 由 W3C 发布一份记录
- 由 W3C 创建一个工作组
- 由 W3C 发布一份工作草案
- 由 W3C 发布一份候选的推荐
- 由 W3C 发布一份被提议的推荐
- 由 W3C 发布推荐

在本教程下面的章节，总结了 HTML、CSS、XML、XSL 在 W3C 的相应活动，包括每项 Web 标准的状态和时间线。

W3C 提交 (W3C Submissions)

任何 W3C 的成员都可向联盟提交希望成为 Web 标准的某项建议（案）。大多数W3C推荐都发源于向联盟做出的某个提交。

如果某项提交在 W3C 的工作领域（或宪章）内，那么 W3C 将决定是否启动对该项提议的改进工作。

W3C 记录 (W3C Notes)

通常，一项对 W3C 的提交会成为一份记录。记录是对作为一份公共文档来提炼的一项提议的描述。

W3C 仅把记录用户讨论。记录的发布并不代表对其的认可。记录的内容是由提交此记录的会员来编辑的，而不是 W3C。记录可在任何时间被更新、替换或废弃。记录的发布也不表明 W3C 已启动与此记录相关的任何工作。

W3C 工作组 (W3C Working Groups)

当某项提交被 W3C 承认，一个工作组就会成立，其中包括会员和其他有兴趣的团体。

工作组通常会定义一个时间表，并发布有关被提议标准的工作草案。

W3C 工作草案 (W3C Working Drafts)

W3C 工作草案通常会被发布于 W3C 的网站上，连同对公共注解的邀请。

工作草案会说明进行中的工作，但不应被用作任何参考材料。其内容可在任何时间被更新、替换或废弃。

W3C 候选推荐 (W3C Candidate Recommendations)

某些规范会比其他规范更复杂，并可能需要来自会员和软件开发商的更多的经费、更多时间以及更多测试。有时，这些规范会作为候选的推荐来发布。

候选的推荐也是一种"正在进行的工作"，同样不应被用作参考材料。此文档可在任何时间被更新、替换或废弃。

W3C 提议推荐 (W3C Proposed Recommendations)

提议的推荐意味着工作组中工作的最后阶段。

提议推荐也是一种"正在进行的工作"。此文档可在任何时间被更新、替换或废弃。不过即使它不意味着 W3C 的任何官方的认可，在极多的情况下，提议的推荐无论在内容还是时间上都已接近于最后的推荐。

W3C 推荐 (W3C Recommendations)

W3C 推荐已经通过了 W3C 会员们的评审，并得到了 W3C 主任的正式批准。

W3C 推荐是一份稳定的文档，并可被用作参考材料。

在本教程下面的章节，总结了 HTML、CSS、XML、XSL 在 W3C 的相应活动，包括每项 Web 标准的状态和时间线。

参考手册

万维网联盟（World Wide Web Consortium，W3C）

W3C HTML 活动

HTML 是 Web 上的通用标记语言。

HTML 教程

如需学习更多有关 HTML 的内容，请阅读我们的 HTML 教程。

HTML 版本

HTML 2.0

HTML 2.0 是 1996 年由 Internet 工程工作小组的 HTML 工作组开发的。

HTML 2.0 是过时的 HTML 版本。目前在市场上可以找到的浏览器都依赖于更新版本的 HTML。对于一位 WEB 开发者而言，没有任何必要需要 HTML 2.0 标准。

HTML 3.2

HTML 3.2 作为 W3C 标准发布于 1997 年 1 月 14 日。HTML 3.2 向 HTML 2.0 标准添加了被广泛运用的特性，诸如字体、表格、applets、围绕图像的文本流，上标和下标。

这些被添加到 1997 年 HTML 3.2 标准的元素之一 - <font> 标签 - 为 HTML 内容和呈现的分离这个重要的任务带来了不必要的麻烦。

HTML 4.0

作为一项 W3C 推荐，HTML 4.0 被发布于 1997 年 12 月 18 日。而仅仅进行了一些编辑修正的第二个版本发布于 1998 年 4 月 24 日。

HTML 4.0 最重要的特性是引入了样式表（CSS）。

我们的 W3C CSS 章节总结了 W3C CSS 活动。

HTML 4.01

作为一项 W3C 推荐，HTML 4.01 发布于 1999 年 12 月 24 日。

HTML 4.01 是对 HTML 4.0 的一次较小的更新，对后者进行了修正和漏洞修复。

W3C 不会继续发展 HTML。未来 W3C 的工作会集中在 XHTML 上。

XHTML 1.0 (最新版本的HTML)

XHTML 1.0 使用 XML 对 HTML 4.01 进行了重新地表示。

作为一项 W3C 推荐，XHTML 1.0 发布于 2000 年 1 月 20 日。

我们的 W3C XHTML 章节总结了 W3C XHTML 活动。

HTML 5

W3C 于 2008 年 1 月 22 日发布 HTML 5 工作草案。

通过制定如何处理所有 HTML 元素以及如何从错误中恢复的精确规则，HTML 5 改进了互操作性，并减少了开发成本。

HTML 5 中的新特性包括了嵌入音频、视频和图形的功能，客户端数据存储，以及交互式文档。

HTML 5 还包含了新的元素，比如：<nav>, <header>, <footer> 以及 <figure> 等等。

HTML 5 工作组包括：AOL, Apple, Google, IBM, Microsoft, Mozilla, Nokia, Opera, 以及数百个其他的供应商。

W3C HTML 规范和时间线

规范	推荐
HTML 3.2	1997 年 1 月 14 日
HTML 4.0	1998 年 5 月 24 日
HTML 4.01	1999 年 12 月 24 日
HTML 5	2010 年 6 月 24 日（最新草案）

在下一节会找到有关 XHTML 规范和时间线的内容。

W3C 参考手册:

W3C HTML 首页

W3C XHTML 活动

XHTML 是更严谨更纯净的 HTML 版本。

XHTML 教程

如需了解如何把网站转换至 XHTML，请阅读我们的 XHTML 教程。

XHTML 版本

XHTML 1.0

作为一项 W3C 推荐，XHTML 1.0 发布于 2000 年 1 月 26 日。

XHTML 1.0 Revision

作为一项 W3C 推荐，XHTML 1.0 第二版发布于 2002 年 8 月 1 日。它不是一个新的版本，而是一次更新和漏洞修复。

XHTML 1.1

作为一项 W3C 推荐，XHTML 1.1 发布于 2001 年 5 月 31 日。

XHTML 1.1 第二版

作为一项 W3C 推荐，XHTML 1.1 (SE) 发布于 2010 年 11月 23日。

关于 XHTML 1.0

XHTML 1.0 是自 1997 年以来对 HTML 的第一次主要的改变，同时也是在向更广泛的用户代理提供更丰富网页的道路上迈出的非常重要的一步，这些用户代理（代理）包括桌面电脑、移动设备和手机等等。

XHTML 是一项可从 HTML 4.01 平稳迁移的 XML 应用。W3C 把 HTML 4.01 重构为 XML 的第一个步骤，导致了 XHTML 1.0 的诞生。XHTML 1.0 依赖于 HTML 4.01 标签所提供的语义。

下一步是把 XHTML 模块化为更小的元素集合，使得 XHTML 和其他标记语言（比如矢量图形和多媒体）的结合更加容易。

同时，XHTML 的模块化还可以减少开发费用，改善与其它应用程序（比如数据库）的协同，更易与不同的用户代理（浏览器）进行通信，以及 HTML 和不同 XML 标准之间更纯净的整合。

W3C XHTML 活动

XHTML 1.0

XHTML 1.0 是使用 XML 对 HTML 4.01 进行的重新表示。

如需学习更多有关 XHTML 的知识，请访问我们的 XHTML 教程。

XHTML 1.1 (模块化的 XHTML)

小型设备（比如移动电话）无法支持 XHTML 的全部功能。XHTML 1.1 将规范划分为具备有限功能的模型。小型浏览器可以通过支持选定的模型来减低其复杂性（不过一旦选定某个模型，就必须支持其全部特性）。

XHTML 1.1 是一门严格的语言。 XHTML 1.1 不能向后兼容 HTML 4。

XHTML 基础

XHTML Basic 是 XHTML 1.1 的小型子集。它仅包含基本的 XHTML 特性，比如文本结构、图像、基本的标单以及基本的表格。它是为小型浏览器设计的（比如在手持设备中）。

XHTML 事件

正是由于 XHTML 中对 W3C 文档对象模型级别 2 的支持，事件处理器就可以依附在 XHTML 元素上，这样父元素就可以在子元素之前或之后来处理事件。

如需学习更多有关 DOM 的知识，请学习我们的 DOM 教程。

XHTML 打印

XHTML-Print 是 XHTML 1.1 (模块化的 XHTML) 的一部分。

XHTML-Print 被设计用于移动设备和廉价的打印机，这些设备通常可在没有打印缓存和为设备定制的打印驱动的情况下，将一张页面从头到尾打印出来。

XForms

通过 XHTML 表单，用户可以访问某张页面，向页面添加信息，然后向Web服务器提交页面。

XForms 是 HTML 表单的继任者，提供一种更完善且独立于呈现的 Web交 互事务处理方式。用于它被设计为与 XHTML 进行整合，我们期望未来的电子商务应用程序会需要需要 XForms。

XHTML 模块化

XHTML 模块化指的是，把 XHTML 1.0 划分为可提供特定功能的小型模型的集合。

XHTML 1.0 的模块化是通过使用 XML DTD (Document Type Definition) 来实现的。

XHTML 2.0 的模块化是通过使用 XML Schemas 来实现的。

如需学习更多有关 DTD 的知识，请学习我们的 DTD 教程。

如需学习更多有关 XML Schemas 的知识，请学习我们的 XML Schemas 教程。

XHTML 2.0

XHTML 2.0 是下一代的标记语言。其功能性预计和 XHTML 1.1 很相似，但是可能被变更来遵守 XML 标准的要求，比如 XML Linking 和 XML Schema。

XLink

XLink 是在 XML 文档中创建超链接的一门语言。XLink 与 HTML 链接很相似 - 但是更加强有力地支持简单链接（比如 HTML）和扩展链接（用于把多项资源链接到一起）。

您可以在我们的 XLink 教程 中学习更多有关 XLink 的知识。

HLink

HLink 增加了一项能力，可规定在 XHTML 中元素哪项元素可表示超链接，并规定如何对超链接进行遍历。

HLink 是对 XLink 的扩展。

W3C HTML 规范 和 时间线

规范	草案/提议	推荐
XHTML 1.0	2000 年 1 月 26 日	
XHTML 1.0 修订版	2002 年 8 月 1 日	
XHTML 1.1	2001 年 5 月 31 日	
XHTML Modules	2001 年 4 月 10 日	
XHTML Modules 1.1	2008 年 10 月 8 日	
XHTML Basic	2000 年 12 月 19 日	
XHTML Basic 1.1	2008 年 7 月 29 日	
XHTML Events	2003 年 10 月 14 日	
XHTML Print	2006 年 9 月 20 日	
XHTML Media Types (SE)	2009 年 1 月 16 日	
XHTML 2.0	2006 年 7 月 26 日	
XForms 1.0	2003 年 10 月 14 日	
XForms 1.0 (Third Edition)	2007 年 10 月 29 日	
XForms 1.1	2009 年 10 月 20 日	
XLink	2001 年 6 月 27 日	
HLink	2002 年 9 月 13 日	

W3C 参考手册：

W3C HTML 主页

W3C XML 活动

XML 被设计用来描述、存储、传送及交换数据。

XML 教程

如需学习更多有关 XML 的知识，请阅读我们的 XML 教程。

XML 版本

XML 1.0

作为一项 W3C 推荐，XML 1.0 发布于 1998 年 2 月 10 日。

XML 1.0 (第二版)

作为一项 W3C 推荐，XML 1.0 (SE) 发布于 2000 年 10 月 6 日。

第二版仅仅是在合并第一版的勘误表的基础上进行的修正（漏洞修复）。

XML 1.0 (第三版)

第二版仅仅是在合并第一版和第二版的勘误表的基础上进行的修正（漏洞修复）。

XML 1.1

作为一份工作草案，XML 1.1 发布于2001 年 12 月 13 日，并作为一项候选推荐发布于2002年10月15日。

XML 1.1 允许在名称中使用几乎所有的 Unicode 字符。

其他 W3C XML 技术

XML 命名空间（Namespaces）

XML 命名空间可规定一种方法，通过与 URI 引用相关联的方式，来定义在 XML 中使用的元素和属性名称。

XML Linking (XLink、XPointer 以及 XML Base)

XML Linking 语言 (XLink)，允许您向XML文档中插入链接。

XML Pointer 语言 (XPointer)，允许将地址链接到 XML 文档的具体部分。

XML Base 是一种用于对外部 XML 资源进行默认引用的标准。(与 HTML 中的 <base> 类似)。

XInclude

XInclude 是一种使用元素、属性以及 URI 引用来合并 XML 文档的机制。

W3C XML 规范和时间线

规范	草案/提议	推荐
XML 1.0	1998 年 2 月 10 日	
XML 1.0 (2.Ed)	2000 年 10 月 6 日	
XML 1.0 (3.Ed)	2004 年 2 月 4 日	
XML 1.1	2004 月 2 月 4 日	
XML 1.1 (2.Ed)	2006 年 8 月 16 日	
XML 1.0 Namespaces	1999 年 1 月 14 日	
XML 1.0 Namespaces SE	2004 年 3 月 4 日	
XML 1.1 Namespaces	2004 年 3 月 4 日	
XML 1.1 Namespaces SE	2006 年 8 月 16 日	
XML Infoset	2001 年 10 月 24 日	
XML Infoset (2.Ed)	2004 年 2 月 4 日	
XML Base	2001 年 6 月 27 日	
XLink 1.0	2001 年 6 月 27 日	
XPointer Framework	2003 年 3 月 25 日	
XPointer element() scheme	2003 年 3 月 25 日	
XPointer xmlns() scheme	2003 年 3 月 25 日	
XInclude 1.0	2004 年 12 月 20 日	
XInclude 1.0 SE	2006 年 11 月 15 日	
XML Processing Model	2004 年 4 月 5 日	
XMLHttpRequest Object	2010 年 8 月 3 日	

W3C 参考手册:

W3C XML 首页

W3C CSS 活动

CSS (Cascading Style Sheets)可描述文档如何被显示。

CSS 教程

如需学习更多有关 CSS 的知识，请阅读我们的 CSS 教程。

CSS1

作为一项 W3C 推荐，CSS1 发布于 1996 年 12 月 17 日。1999 年 1 月 11 日，此推荐被重新修订。

CSS2

作为一项 W3C 推荐，CSS2 发布于 1999 年 1 月 11 日。CSS2 添加了对媒介（打印机和听觉设备）和可下载字体的支持。

CSS3

CSS3 计划将 CSS 划分为更小的模块。

CSS4

从CSS3开始，模块将被单独定义，例如 css4选择器可以在CSS3模块定义。

CSS 属性

W3C 定义了以下CSS属性：

- 打印属性
- 移动设备属性
- TV 属性

W3C CSS 规范和时间线

规范	草案/提议	推荐日期
CSS 1	17. Dec 1996	
CSS 1 (Revised)	11. Apr 2008	
CSS 2	12. May 1998	
CSS 2.1	07. Jun 2011	
DOM Level 2 Style Specification	13. Nov 2000	
CSS Style Attributes	12. Oct 2010	
CSS 3 Namespaces Module	29. Sep 2011	
CSS 3 Selectors	29. Sep 2011	
CSS 3 User Interface Module	17. Jan 2012	
CSS 3 Fonts Module	12. Feb 2013	
CSS 3 Color Module	07. Jun 2011	
CSS 3 Backgrounds and Borders Module	24. Jul 2012	
CSS 3 Text Module	13. Nov 2012	
CSS 3 Lists and Counters Module	24. May 2011	
CSS 3 Line Module	15. May 2002	
CSS 3 Basic Box model	09. Aug 2007	
CSS 3 Multi-column Layout Module	12. Apr 2011	
CSS 3 Ruby Module	30. Jun 2011	
CSS 3 Speech Module	20. Mar 2012	
CSS 3 Paged Media Module	10. Oct 2006	
CSS 3 Generated content for Paged Media Module	29. Nov 2011	
CSS 3 Values and Units Module	28. Aug 2012	
CSS 3 Cascading and Inheritance	03. Jan 2013	
CSS 3 Template Layout Module	29. Nov 2011	
CSS 3 Media Queries	19. Jun 2012	
CSS Mobile Profile 2.0	10. Dec 2008	
CSS TV Profile 1.0	14. May 2003	
CSS Print Profile 1.0	13. Oct 2006	

W3C 参考手册:

W3C CSS 主页

W3C XSL 活动

W3C开始发展XSL，因为有一个基于XML的样式表语言的需要。

XSL 语言包括三部分：XSLT、XPath 以及 XSL 格式化对象。

XSL 教程

如需学习更多有关 XSL 的知识，请阅读我们的 XSL 教程。

XSL 版本

XSL 1.0

作为一项 W3C 推荐标准，XSL 1.0 作为一门表达样式表的语言被发布于 2001 年 10 月 15 日。它由三部分组成：XSLT、XPath 以及 XSL 格式化对象。

XSLT 1.0

XSLT 1.0于 1 999年11月16日成为 W3C 推荐标准。XSLT 是一门用于把 XML 文档转换为其他 XML 文档的语言。

XSLT 2.0

XSLT 2.0于 2007 年 1 月 23 日成为 W3C 推荐标准。

XSL-FO (XSL 格式化对象)

XSL 格式化对象一个用于规定格式化语义的词汇表。格式化指的是把XSL转换的结果转变为适合阅读器或收听器的过程。虽然不存在针对 XSL 格式化对象的独立 W3C 文档，但是还是可以在 XSL 1.0 推荐标准中找到相关的描述。

W3C XSL 规范和时间线

规范	草案/提议	推荐时间
XSL 1.0	15. Oct 2001	
XSL 1.1	05. Dec 2006	
XSLT 1.0	16. Nov 1999	
XSLT 2.0	23. Jan 2007	
XSLT 2.0 Requirements	14. Feb 2001	

W3C 参考手册:

W3C XSL 主页

W3C XML Schema 活动

XML Schema 是基于 XML 的 DTD 替代物。

XML Schema 教程

如需学习更多有关 XML Schema 的知识，请阅读我们的 XML Schema 教程。

XML Schema

XML 1.0 支持可定义文档结构的 DTD。

XML Schema 对应用程序、文档结构、属性和数据类型有着更良好的支持。

未来的 XML 版本有赖于 XML Schema 来定义 XML 文档的类型。

- XML Schema 的结构（XML Schema Structure）规定了 XML Schema 的定义语言。
- XML Schema 的数据类型为 XML 规定了可扩展的数据类型。

W3C XML 规范和时间线

规范	草案/提议	推荐时间
XML Schema (XSD)	02. May 2001	
XML Schema 1.0: Structures	02. May 2001	
XML Schema 1.0: Datatypes	02. May 2001	
XML Schema (2.Ed)	28. Oct 2004	
XML Schema Structures (2.Ed)	28. Oct 2004	
XML Schema Datatypes (2.Ed)	28. Oct 2004	
XML Schema Component Designators	19. Jan 2010	
XML Schema 1.1: Structures	21. Jul 2011	
XML Schema 1.1: Datatypes	21. Jul 2011	

W3C 参考手册:

W3C XML Schema 首页

W3C XPath 活动

XPath是一门用于选取 XML 文档的部件的语言。

XPath 被设计为供 XSLT、XQuery 以及 XPointer 使用。

教程

如需学习更多有关 XPath 的知识，请阅读我们的 XPath 教程。

如需学习更多有关 XQuery 的知识，请阅读我们的 XQuery 教程。

如需学习更多有关 XSLT 的知识，请阅读我们的 XSLT 教程。

XPath 版本

XPath 1.0

XPath 1.0 于 1999 年 11 月 16 日成为 W3C 推荐标准。

XPath 2.0

XPath 2.0 于 2007 年 1 月 23 日成为 W3C 推荐标准。

XPath 2.0 是一门由 XPath 1.0 和 XQuery 衍生而来的语言。XPath 2.0 和 XQuery 1.0 的产生是同源的，它们拥有不少相同的语法，而且不少文本也是一致的。

W3C XSL 规范和时间线

规范	草案/提议	推荐时间
XPath 1.0	16. Nov 1999	
XPath 2.0 Requirements	03. Jun 2005	
XPath 2.0 Language	14. Dec 2010	
XPath 2.0 Functions	14. Dec 2010	
XPath 2.0 Data Model	14. Dec 2010	
XPath 2.0 Semantics	14. Dec 2010	
XPointer	16. Aug 2002	

W3C 参考手册:

W3C XSL 首页

W3C XQuery 活动

XQuery 是一门用于从 XML 文档中提取数据的语言。

教程

如需学习更多有关 XQuery 的知识，请阅读我们的 XQuery 教程。

XQuery 版本

XQuery 1.0

XQuery 语言支持从 XML 文档提取数据的查询工具。

W3C XQuery 规范和时间线

规范	草案/提议	推荐时间
XQuery Requirements	23. Mar 2007	
XQuery Use Cases	23. Mar 2007	
XQuery 1.0	14. Dec 2010	
XQuery 1.0 Functions	14. Dec 2010	
XQuery 1.0 Data Model	14. Dec 2010	
XQuery 1.0 Semantics	14. Dec 2010	
XQueryX	14. Dec 2010	
XQuery 3.0 Requirements	16. Sep 2010	
XQuery 3.0 Use Cases	14. Dec 2010	
XQuery 3.0	14. Jun 2010	

W3C 参考手册:

W3C XSL 首页

W3C DOM Activities

文档对象模型 (DOM：Document Object Model) 是一个平台，一个中立于语言的应用程序编程接口 (API)，允许程序访问并更改文档的内容、结构和样式。

DOM Tutorials

如需学习更多有关 DOM 的知识，请阅读我们的 HTML DOM 教程和 XML DOM 教程。

DOM 级别 0

DOM 级别 0 不是 W3C 规范。而仅仅是对在 Netscape Navigator 3.0 和 Microsoft Internet Explorer 3.0 中的等价功能性的一种定义。

DOM 发展过程中的关键角色有：ArborText、IBM、Inso EPS、JavaSoft、Microsoft、Netscape、Novell、the Object Management Group、SoftQuad、Sun Microsystems 以及 Texcel。

W3C 的 DOM 级别 1 建立于此功能性之上。

DOM 级别 1

DOM 级别 1 专注于 HTML 和 XML 文档模型。它含有文档导航和处理功能。

DOM 级别 1 于 1998 年 10 月 1 日成为 W3C 推荐标准。

第二版的工作草案在 2000 年 9 月 29 日。

DOM 级别 2

DOM 级别 2 对 DOM 级别 1 添加了样式表对象模型，并定义了操作附于文档之上的样式信息的功能性。

DOM 级别 2 同时还定义了一个事件模型，并提供了对 XML 命名空间的支持。

作为一项 W3C 推荐标准，DOM 级别 2 规范发布于 2000 年 11 月 13 日：

DOM Level 2 核心

DOM Level 2 核心 规定了访问和更改文档内容及结构的一个 API，此 API 同时包含用于 XML 的接口。

DOM Level 2 HTML

DOM Level 2 HTML 规定了操作 HTML 文档结构和内容的 API。（这部分规范仍然是工作草案）

DOM Level 2 Views

DOM Level 2 规定了对文档视图进行访问和更改的 API。视图是与原文档相关联的表现形式或某种备用的表现形式。

DOM Level 2 Style

DOM Level 2 Style 规定了动态访问及更改内容样式表的 API。

DOM Level 2 Events

DOM Level 2 Events 规定了访问文档事件的 API。

DOM Level 2 Traversal-Range

DOM Level 2 Traversal-Range 规定了动态遍历和识别文档中内容范围的 API。

DOM 级别 3

DOM Level 3 规定了内容模型 (DTD 和 Schemas) 和文档验证。同时规定了文档加载和保存、文档查看、文档格式化和关键事件。DOM Level 3 建立于 DOM Core Level 2 之上。

DOM Level 3 Requirements

DOM Requirements 文档已经为 Level 3 requirements 进行了更新，并于 2000 年 4 月 12 日发布为工作草案。

下面的 DOM Level 3 工作草案发布于 2000 年 9 月 1 日：

DOM Level 3 Core

DOM Level 3 Core 规定了访问和更改文档内容、结构及样式的一个 API。

DOM Level 3 Events

通过增加新的接口和新的事件集，DOM Level 3 Events API 对 Level 2 Event API 的功能进行了扩展。

DOM Level 3 Load and Save

DOM Level 3 Content Model 规定了用于内容加载和保存、内容模型 (DTD and Schemas) 和文档验证支持的 API。

DOM Level 3 Views and Formatting

DOM Level 3 Views 规定了对文档视图进行访问和更改的 API。视图是与原文档相关联的表现形式或某种备用的表现形式。

W3C DOM 规范和时间线

规范	草案/提议	推荐
DOM Level 1	01. Oct 1998	
DOM Level 1 (2.Ed)	29. Sep 2000	
DOM Level 2 Core	13. Nov 2000	
DOM Level 2 HTML	09. Jan 2003	
DOM Level 2 Views	13. Nov 2000	
DOM Level 2 Style	13. Nov 2000	
DOM Level 2 Events	13. Nov 2000	
DOM Level 2 Traversal-Range	13. Nov 2000	
DOM Level 3 Requirements	26. Feb 2004	
DOM Level 3 Core	07. Apr 2004	
DOM Level 3 Events	31. May 2011	
DOM Level 3 Load and Save	07. Apr 2004	
DOM Level 3 Validation	27. Jan 2004	
DOM Level 3 XPath	26. Feb 2004	
DOM Level 3 Views	26. Feb 2004	

W3C 参考手册:

W3C DOM 首页

W3C Soap 活动

SOAP 是一种简单的基于 XML 的协议，它使应用程序通过 HTTP 来交换信息。

或者简单的说： SOAP 是基于 XML 的 Web Services 间的通信协议。

SOAP

SOAP (Simple Object Access Protocol) 是一种中立于平台和语言的轻量级通信协议，使得程序可以通过标准的因特网 HTTP 进行通信。

如需学习更多有关 SOAP 的知识，请阅读我们的 SOAP 教程。

SOAP 1.1

在 2000 年 5 月，SOAP 1.1 曾在一份记录中被建议到 W3C（由开发商：IBM, Lotus, Microsoft 以及 Userland），作为用于在分布式环境中交换信息的一种协议。

W3C SOAP 1.1 文档仅仅是一份用于讨论的记录（NOTE）。此记录的发布不代表 W3C 对其任何程度的认可。

SOAP 1.2

W3C 的 XML Protocol 工作组目前正工作于 SOAP 1.2

第一份工作草案发布于 2001 年 12 月 17 日。

SOAP 1.2 于 2003 年 6 月 24 日被发布为 W3C 推荐标准。

W3C SOAP 规范和时间线

规范	草案/提议	推荐时间
SOAP 1.2 Primer	24. Jun 2003	
SOAP 1.2 Primer (2.Ed)	27. Apr 2007	
SOAP 1.2 Messaging	24. Jun 2003	
SOAP 1.2 Messaging (2.Ed)	27. Apr 2007	
SOAP 1.2 Adjuncts	24. Jun 2003	
SOAP 1.2 Adjuncts (2.Ed)	27. Apr 2007	
SOAP 1.2 Test Collection	24. Jun 2003	
SOAP 1.2 Test Collection (2.Ed)	27. Apr 2007	
SOAP 1.2 Attachments	08. Jun 2004	
SOAP 1.2 Email Bindings	03. Jul 2002	
SOAP 1.2 Normalization	08. Oct 2003	
SOAP 1.2 Serialization	08. Jun 2004	
Web Services Addressing 1.0 - Core	09. May 2006	
Web Services Addressing 1.0 - SOAP	09. May 2006	

W3C 参考手册:

W3C SOAP 首页

W3C WSDL 活动

WSDL 是基于 XML 的用于描述 Web Services 以及如何访问 Web Services 的语言。

WSDL

WSDL (Web Services Description Language) 是一种用于描述 Web Services 的 XML 格式。

如需学习更多有关 WSDL 的知识，请阅读我们的 WSDL 教程。

WSDL 1.1

作为一种可描述 Web Services 的 XML 格式，WSDL 1.1 于 2001 年 3 月曾在一份记录中被建议到 W3C（由Ariba、IBM 以及 Microsoft）。

此记录还描述了如何结合 SOAP 1.1、HTTP GET/POST 以及 MIME 来使用 WSDL。

W3C WSDL 1.1 仅仅是用于讨论的记录（NOTE）。此记录的发布不代表 W3C 对其任何程度的认可。

WSDL 1.2

第一份工作草案发布于 2001 年 12月 17 日。

最新的工作草案发布于 2003 年 6月 11 日。

WSDL 2.0

W3C 的 XML Protocol 工作组目前正在工作于 WSDL 2.0。

WEB 结构

Web结构的核心是一台Web 服务器，它一般由一台独立的服务器承担，数据库服务器为信息管理系统数据库服务器，各客户机数据请求均由Web服务器提交给数据库服务器，再由Web服务器返回发给请求的客户机。

这里的Web服务器可设为一个网关，一端接信息管理系统的内部网，另一端接入企业Intranet，这样既避免了内部网直接暴露于外部，又使内部都可访问到Web站点。

W3C WSDL 规范和时间线

规范	草案/提议	推荐时间
WSDL 1.1 Note	15. Mar 2001	
WSDL Usage Scenarios	04. Jun 2002	
WSDL Requirements	28. Oct 2002	
WSDL Architecture	11. Feb 2004	
WSDL Glossary	11. Feb 2004	
WSDL Usage Scenarios	11. Feb 2004	
WSDL 1.2 Core Language	11. Jun 2003	
WSDL 1.2 Message Patterns	11. Jun 2003	
WSDL 1.2 Bindings	11. Jun 2003	
WSDL 2.0 Primer	26. Jun 2007	
WSDL 2.0 Core Language	26. Jun 2007	
WSDL 2.0 Adjuncts	26. Jun 2007	
WSDL 2.0 SOAP 1.1 Binding	26. Jun 2007	
WSDL 2.0 RDF Mapping	26. Jun 2007	
Web Services Addressing Core	09. May 2006	
Web Services Addressing SOAP Binding	09. May 2006	
Web Architecture	15. Dec 2004	

W3C 参考手册:

W3C Web Services 首页

W3C RDF and OWL 活动

RDF 和 OWL 是两项重要的语义网技术。

语义网 (Semantic Web)

语义网是为资产管理、企业整合及网络数据的共享和重用提供的一个框架。

语义网为企业、应用程序、公司、团体和个人间的数据共享提供了一个独立于平台和软件的框架。

RDF 和 OWL 是的语义网的关键技术。它们分别阐述了结构性描述和以万维网为基础的本体论。

RDF - 资源描述框架（Resource Description Framework）

RDF 是一门向万维网表达信息的语言。

RDF 可用于描述 web 资源，比如标题、作者以及版本信息、内容描述、可用时间表等等。

如果您需要学习有关 RDF 的知识，请访问我们的 RDF 教程。

OWL - Web 本体语言

OWL 是用于定义本体的语言。

本体可描述知识的领域。可供人类或软件用来分享有关对象的信息，这些对象可以是汽车、房屋、机器、书籍、产品、金融交易等等。

OWL 被设计为用于对信息进行处理（而不是现实信息）。

假如您需要学习更多有关 OWL 的知识，请访问我们的 RDF 教程。

SPARQL - 针对 RDF 的查询语言(Query Language for RDF)

SPARQL 是用于 RDF 数据的标准查询语言，可向开发者提供编写跨越 WEB 上广域 RDF 信息查询程序的途径。

W3C 规范和时间线

规范	草案/提议	推荐时间
RDF Primer	10. Feb 2004	
RDF Test Cases	10. Feb 2004	
RDF Concept	10. Feb 2004	
RDF Semantics	10. Feb 2004	
RDF Schema	10. Feb 2004	
RDF 语法	10. Feb 2004	
OWL Overview	10. Feb 2004	
OWL Guide	10. Feb 2004	
OWL 参考手册	10. Feb 2004	
OWL 语法	10. Feb 2004	
OWL Test Cases	10. Feb 2004	
OWL Use Cases	10. Feb 2004	
Parsing OWL in RDF	21. Jan 2004	
SPARQL Language	15. Jan 2008	

W3C 参考手册

W3C 语义网活动

其他 W3C 活动

本节概况了其他一些重要和有趣的 W3C 活动。

The Web Accessibility Initiative (WAI)

WAI 定义了如何使残障人士更易使用 Web 内容的指导方针。

WAI 通过技术、指导方针、工具、教育、研究以及开发项目，为 "Web accessibility for all" 这个目标而努力。

W3C WAI 推荐标准

数学标记语言 - Mathematical Markup Language (MathML)

MathML 是一项用于描述数学符号的 XML 标准。

MathML 的目标的使数学能够在 Web 上被提供、接受和处理，就像 HTML 可以令文本实现的功能一样。

W3C MathML 推荐标准

可缩放的矢量图形 - Scalable Vector Graphics (SVG)

SVG 是一门用于在 XML 中描述二位图形的语言。

SVG 运行三种类型的图形对象：矢量图形形状、图像和文本。

特性设置包括了变换、裁剪路径、alpha 遮罩、滤镜效果、模版对象以及可扩展性。

W3C SVG 1.1 推荐标准

W3C SVG 1.2 工作草案

墨水标记语言 - Ink Markup Language (InkML)

InkML 是用于表达数字墨水数据的 XML 数据格式，这类数据的输入是通过作为多通路系统组成部分的电子笔或输入笔。

W3C InkML 工作草案

国际化 - Internationalization

W3C 国际化活动的目标是，在 W3C 内部以及与其他组织一起，建议并调整任何技术、协定、指导方针和活动，使得在不同的语言、脚本和文化范畴内更易在全球范围内使用 W3C 技术。

W3C 国际化活动

语音浏览器活动

W3C 的语音浏览器活动的工作是使人们可以通过口语指令和语音合成进行交互，以扩展 Web（的使用范畴）。

语音可扩展标记语言 （VoiceXML）

语音合成标记语言 （VoiceXML）

语音识别语法规范

语音浏览器呼叫控制 （CCXML）

W3C 参考手册:

W3C 首页

Web 品质

Web 品质 - 标准

根据web标准编写您的页面有助于改善您的网站品质。

HTML 标准

XHTML 是最新的 HTML 标准，是用 XML 重新表达的 HTML 4.01。

根据 HTML 4.01 编写页面可使您的站点尽可能地接近 XHTML 标准。

阅读更多有关 HTML 的内容。

CSS 标准

对于高品质的站点来说，使用层叠样式表（CSS）是将内容与样式分离的首选方式。通过使用 CSS，您能够在一个单独的文档中存储有关页面样式的所有信息。

所有现代的 web 浏览器均支持 CSS 1 和 CSS 2 标准。

对于不同的浏览器，使用 CSS 都可以改进网站的品质，并提高可读性。同时还可以极大地减少您的网站开发成本。

阅读更多关于 CSS 的内容。

Web 验证

web 验证工具是一种软件程序，可根据 web 标准对您的网站进行检查。

当您使用验证工具检查过HTML, XHTML 或 CSS 文档之后，验证器就会根据您选择的标准返回一系列所发现的错误。通常，验证器会返回所发现错误的行号。

请确保你在发布页面前进行验证成为一种习惯。

阅读更多关于 页面验证 的内容。

WAI - 无障碍网页倡议

WAI 指的是无障碍网页倡议("Web Accessibility Initiative")，是由 W3C 发起的。

WAI 协调全球的组织通过六项主要的工作领域来提升因特网的可用性：技术、指导方针、工具、教育、研究以及开发。

您能够通过根据 WAI 的指导方针编写的页面，来改善网站的品质，并使得您的站点可用于更多人群（及浏览器）。

您将在本教程稍后的章节学习更多有关 WAI 的内容。

Web 品质 - 重要的 HTML 元素

对于提升 web 品质，<DOCTYPE>、<title> 以及 <h1> 都是重要的标签。

<!DOCTYPE> 元素

所有的 HTML 和 XHTML 页面都应当使用 <Doctype> 元素来定义遵照何种 HTML 版本。

doctype 定义了您正在使用的 HTML 版本，并为浏览器提供重要的信息以便其更快速一致地呈现您的页面。

文档类型声明同时也使验证软件可以对页面的语法进行检查：

HTML 5

```
<!DOCTYPE html>
```

HTML 4.01 Strict, Transitional, Frameset

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0 Strict, Transitional, Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

<title> 元素

<title> 元素是最重要的 HTML 元素之一。它的主要功能是描述网页的内容。

即使标题不是网页的一个可见的部分，它对于提升网站的品质依然是重要的，这是因为它在以下位置都是可见的：

- 搜索引擎列表
- 窗口的标题栏
- 用户的书签中

标题应当尽可能地短，并具有可描述性。

当某个用户在 internet 上搜索网站时，大部分搜索引擎都会在搜索结果中显示出网站的标题。请确保标题与网页的内容是吻合的。这样的话用户有更多的可能通过点击这些链接来访问到你的网站。

当用户访问您的网站时，在窗口的标题栏中标题是可见的。请确保即使窗口被最小化，标题同样能起到描述网站内容的作用。

在用户访问你的网站之后，网页的标题会存储于历史文件夹（用户甚至会把网页收藏到他的收藏夹中）。为了后续的成功访问，同样请确保标题可以清楚地描述您的网站。

优秀的标题：

```
&lt;title&gt;HTML Tutorial&lt;/title&gt;

&lt;title&gt;XML Introduction&lt;/title&gt;
```

差距的标题的例子：:

```
&lt;title&gt;Introduction&lt;/title&gt;

&lt;title&gt;Chapter 1&lt;/title&gt;
```

W3Cschool拥有一整套组织良好、易于理解的 HTML、CSS、JavaScript、 DHTML、XML、XHTML、WAP、ASP、SQL 教程，并包含非常多实例和源代码。

<h1> 元素

<h1> 元素用来描述网页中最上层的标题。

由于一些浏览器会默认地把 <h1> 元素显示为很大的字体，因此会有一些 web 开发者使用 <h2> 元素代替 <h1> 元素来显示最上层的标题。这样做不会对读者产生影响，但会使那些试图"理解网页结构"的搜索引擎和其他软件感到迷惑。

请确保把 <h1> 用于最顶层的标题，<h2> 和 <h3> 用于较低的层级。

可以试着根据此模版来构造您的网页：

```
<h1>This is the main heading</h1>

Some initial text

<h2>This is a level 2 heading</h2>

This is some text. This is some text. This is some text.

<h3>This is a level 3 heading</h3>

This is some text. This is some text. This is some text.

<h3>This is a level 3 heading</h3>

This is some text. This is some text. This is some text.
```

如果您不喜欢默认的标题字体尺寸，可以使用样式或样式表来改变。

Web 品质 - 样式表

使用样式表对于提升网页品质至关重要。

不要使用 <font> 标签！

应使用 CSS 来设置显示网页上的字体尺寸。请不要使用 font 标签。

使用 <font> 标签会增加文档的规模，而且使您每次改变标准文字尺寸的工作成为一场梦魇。

请设想一下下面的情况：

一天您决定修改网站中所有标题的颜色和尺寸。通过 CSS，您只需要修改一行就可以做到这一点。假如您使用了 <font>标签，那么您需要把网站中所有页面的所有标题都修改一遍。

使用样式替代 <font> 标签可使我们更轻松地为网页制作高质量的界面。

请勿使用固定的字体尺寸

不要使用固定的尺寸值。请始终使用相对的尺寸值。

这项建议最重要的理由是无法通过浏览器重新调整固定尺寸的大小。

您的访问者会使用不同的设备（显示器）、不同的浏览环境（光线）以及可能的残疾（弱视）。

例如，可以把某人的文字尺寸设置为100%（或者 medium ），主标题设置为140%（或者 x-large ），而次级的标题设置为120%（或者 large ），这样用户就可以使用浏览器来重新设定最喜欢的尺寸了。

注意： 通过调整网页的文本尺寸的功能，也可以改变打印页面的文字数目。

请勿使用很小的默认字体尺寸

一些网站会使用很小的文字尺寸，这样就可以向每张页面"塞"入更多的内容，或者使页面看上去更"时髦"。

再次重申，使用不同的设备（显示器）、不同的浏览环境（光线）以及可能的残疾（弱视），都可能对用户造成阅读障碍。

请不要逼迫用户每次访问你的站点时都要放大文本的尺寸。

始终使用一致的背景颜色

大部分网页都会为不同的文本元素使用颜色。标题和链接的颜色通常与正文的文本颜色是不同的。

作为一位 web 设计者，您应当意识到的事实是，您的访问者能够修改默认的颜色选项。

如果您为 web 元素定义了颜色，那么同样应当定义背景颜色。

如果不定义背景颜色，那么您的网站可能会被糟糕的颜色组合搞砸（比如红色背景上面的亮红文字，或者深色文本搭配的深色背景）。

如果您不规定背景颜色，可能会使文本很难被识别。

Web 品质 - 可读性

正确使用字体和颜色可使您的网站更易阅读。

留意颜色的对比

对于视力不太好的人或者对于不太好的显示设备来说，黑底白字或者白底黑字是最佳的。

在亮色背景上的灰色文字，对比度是很差的：

Grey text on a white background (#EEEEEE)
Grey text on a white background (#CCCCCC)
Grey text on a white background (#AAAAAA)
Grey text on a white background (#888888)
Grey text on a white background (#666666)
Grey text on a white background (#444444)
Grey text on a white background (#222222)
Grey text on a white background (#111111)

在暗色背景上的灰色文字，其对比度同样很差：

Grey text on a black background (#222222)
Grey text on a black background (#444444)
Grey text on a black background (#666666)
Grey text on a black background (#888888)
Grey text on a black background (#AAAAAA)
Grey text on a black background (#CCCCCC)
Grey text on a black background (#DDDDDD)
Grey text on a black background (#EEEEEE)

某些搭配 - 比如黑色和红色，黑色和蓝色，黄色和绿色 - 都会使人产生视觉疲劳：

Black text on a red background
Black text on a blue background
Yellow text on a green background

这些搭配并不糟糕：

Black text on a grey background
Black text on a light blue
Black text on antique white
White text on dark blue

留意字母的间距

对于视力比较弱的读者，比较近的字母间距会带有不小的阅读困难。

文字间距适中的文本就比较容易阅读了。

文字间距少的文字难以阅读。

当心你的行距

以下行距易于阅读

具有良好的行距的文本 更易阅读

以下行距难以阅读

行距小的文本 难以阅读

避免花哨的字体

这字体容易阅读....

这种字体是难以阅读....

尽量少用斜体

普通字体易于阅读。

斜体字体不那么容易阅读。

Web Quality - 无障碍(WAI)

能够被残障人士使用的网站才能称得上一个易用的（易访问的）网站。

残障人士指的是那些带有残疾或者身体不健康的用户。

Web Accessibility Initiative - WAI

WAI （由 W3C 在1997年发起）是一系列计划供 web 开发者、创作者以及设计者使用的指导方针 - 关于如何使内容对残障人士更易用。

这些指导方针的目标是易用性（accessibility），但是也有助于使 web 内容可用于更多的浏览器（语音浏览器、移动电话、手持设备），以及更多工作于困难环境的用户（非手持式的、强光、黑暗、弱视、噪音等）。

WAI 对您的网站来说很重要吗？

是的。

每天都有百万计的残障人士在网上冲浪，并且甚至超过百万计的人们正在使用着不良的浏览器设备，或者工作在困难的环境当中。

假如您的网站缺乏诸如可调节字体尺寸、带有文本描述的图形以及便捷的导航，那些人们就无法访问您的信息。

事实上：您的网站剥夺了这些人的权利。

增强网站易用性的理由还有：

- 可提升网站的美誉度和形象
- 可提升户满意度
- 可增加访问者的数量
- 可增加访问者在站点的停留时间
- 可增加访问者的回访数量
- 可同样为无残疾人士增加可用性
- 可为关闭图形功能的访问者增加可用性
- 可为使用老式设备的人群增加可用性
- 可使您的网站为增长速度最快的人群提供服务：老年人群

使用可调节的字体大小

请使用相对的字体尺寸，这样用户就能够使用浏览器菜单来改变默认的字体尺寸。

你可以阅读吗？

你可以阅读吗？

你可以阅读吗？

你可以阅读吗？

可以在您的浏览器菜单选择"查看 - 文字大小"来 要改变字体大。

使用 "alt" 属性

alt 属性允许你为图像（也可以为其它的元素）提供一条相对应的文字。

实例:



有时候浏览器会无法显示图像。具体的原因有：

- 用户关闭了图像显示
- 浏览器是不支持图形显示的迷你浏览器
- 浏览器是语音浏览器（供盲人和弱视人群使用）

如果您使用了 alt 属性，那么浏览器至少可以显示或读出有关图像的描述。

Web 品质 - 国际化

网络无国界。

网络无国界。

"With the Internet follows an absolute requirement to interchange data in a multiplicity of languages, which in turn utilize a bewildering number of characters."

H. Alvestrand, Internet工程工作小组 (IETF), 1998年1月。

国际字符集

所有的 W3C 标准（自从1996年），包括 HTML、XHTML 和 XML 都定义了一个名为 Unicode (ISO 10646) 内部的内部字符集。

所有现代 web 浏览器都在原生地使用此字符集。而大多数在 internet 上传输的文档并没有使用这个 Unicode 字符集。

正因如此，Internet 客户（浏览器）与 Internet 服务器 之间必须有一种在通信中一致使用字符集的方法。

对每个文档在用的字符集进行标记，对于提升网站的品质来说至关重要。

请始终在 <head> 元素内 使用下面的的元元素：

把 X 替换为你所使用的字符集，比如ISO-8859-1、UTF-8 或者 UTF-16。

国际日期

请不要使用类似 "04-03-02" 的日期格式。

上面的日期可以表示为2004年3月2日，或者2002年3月4日，亦或者2002年4月3日。

国际标准化 (ISO) 为日期定义的国际标准格式是 "yyyy-mm-dd"，yyyy 是年，mm 是月，dd 是日。

如果您使用了 ISO 的格式，那么大多数访问者都能明白你的日期。

职业规划

职业规划提示

九项重要的职业规划提示。

1. 学习的步伐不停止

古人说，活到老，学到老。终身学习应该是您的座右铭。

世界在不断变化，每个人都在寻找各自的事业途径。

您只有保证了足够的技能储备，才能确保能够得到一份足够满意的工作。

为了保证您的职业发展，您应当定期地更新您的技能和知识。

2. 学会问、学会听，学会学习

一个好的倾听者可以习得更多。

多听取来自同事、老板以及上级的声音。您可以从他们的经历中学到更多。

问一些您感兴趣的话题，然后听听他们怎么说。让他们告诉您事业如何运作，以及如何可以做得更好。

大多数都是乐意帮助您的。

3. 为目前的工作全力以赴

您目前的工作可能是开始您职业生涯的最好起点。

从本职工作做起，从现在做起，做好当前的工作，没有保留地尽到自己的职责，证明自己是一名有价值的员工。

您所做的工作终究会得到回报。

4. 构建人际网络

您的下一个职业阶段很可能得益于您的人际网络。

您知道吗，超过50%的工作都是通过关系网络获得的。

如果您拥有一张良好的人际网络，那么它会助您发现未来的职业，开拓新的方向，获得新的机会。

请在新的关系上多花些时间吧，同时请不要忽略对已有关系的保持。

从您的人际网络获得有价值信息的最佳途径之一是，定期地问候您的交际人，他们正在做什么，以及有关其职业的新情况。

5. 识别你的工作

识别真正重要的工作，而不是去假设。

一定要确定你目前所做的工作不是因假设得来的。那样会浪费您很多时间和才华。

当您着手一份新的工作时，一定要和主管聊聊首要的那些工作。如果您无法确定哪方面是重要的，就去询问他吧。多谈几次也没关系。您会经常对事实上的重要任务与您所作的假设之间差距感到惊讶。

6. 慎重决定下一个工作

在您开始未来的职业生涯之前，一定要认真考虑您理想中的工作。

您理想的职业应该是什么样的呢？ 最关键的是，您一定要乐在其中。

您是否乐于为其它的同事承担责任？您喜欢和人打交道还是摆弄技术？你希望自己创业吗？您希望成为一位艺术家、一位设计师、一名熟练的工程师，还是一名管理人员？

在您为构建未来的职业生涯之前，请明确您的目标。

7. 为未来做准备

为了明天的梦想，今天就要进行准备。

一刻也不要耽搁。现在就更新您的履历，并且定期持续对其更新。

明天您也许就会看到梦想实现的曙光。为此，您需要准备一份专业的履历，准备好为您的雇主展现潜力无穷的你吧！

如果您不清楚如何写一份履历，或者任何描述自己，请现在就开始学习吧。

8. 量力而行

选择适合个人能力的任务。

您可以通过不同的方式来构建未来的职业生涯。在 W3School 学习是件容易的事情。而获得硕士学位则会困难一些。

您可以通过学习各类型的书籍和教程（比如您在 W3School 所找到的）来为职业添砖加瓦。参加一些带有认证测试的短期培训应该可以为您的履历增加不少分量。同时不要忘了：培养新技术所需要的最具价值的资源是您目前从事的工作。

不要为自己设置不可能完成的任务！

9. 实现您的梦想

把梦想落实为行动！

不要让繁忙的工作扼杀您的梦想。假如您有着更高远的目标，请现在就付诸行动吧！

如果您计划接受更高的教育，获得更好的工作，或者开一间属于自己的公司等等，请不要以日常的工作作为等待的借口。您的日常工作会变得越来越忙，您会陷入激烈的竞争中，并耗尽自己的能量。

如果您此刻就存有能量，那么现在就使用它去实现您的梦想吧！

职业履历（CV）

履历（**CV**）是向雇主推销自己的**“**广告**”**。

什么是 CV ？

- CV 指的是 "Curriculum Vitae"
- Curriculum vitae 在拉丁语中的意思是“生命的故事”
- CV 经常被称为 "Resume"

一份 CV 中包含哪些内容？

一份 CV 至少要包含下列内容：

- 个人信息
- 工作经历
- 技能
- 教育水平
- 个人简介和兴趣
- 推荐

您的个人信息

个人信息应该包括姓名、住址、电话和电子邮件。我建议您把这些信息放到 CV 的顶部，让它看上去像信笺的抬头。

您的工作经历

列出你做过的工作 - 在开头列出您最近的工作经历。

以及简短的工作描述和您的职责。

确保工作经历位于 CV 的第一页。这些些信息概括了您的技能和优势。其它附加的信息应当位于后面的位置。

您的技能

技能最好使用列表来描述。

列出您的技能 - 最重要的和最相关的。

您的教育水平

教育最好使用列表来描述。

列出您学习的内容 - 在开头列出您最近所受的教育。

不要忘记学科选项、特殊的项目、课程或者荣誉证书。

您的推荐

列出一些人的名字 - 比如您学习所在学校的教授、公司的主管 - 确保可以很容易地联系到他们，并且他们愿意为你作积极的推荐。

您的个人简介

个人简介应包括有关年龄、兴趣爱好以及其它相关信息的的附加信息，且这些信息有助于塑造您的正面形象。我建议您把这些内容放到 CV 的最后。

由于这些内容可以展现您的特质，所以雇主会有兴趣了解这些内容。但需要小心，不要过度地描述您的兴趣，也不要描述那些可能影响到工作的兴趣。假设您正在为一个足球队作教练，不要写赢得比赛的次数。如果他们感兴趣的话，让他们在以后的面试中与您聊这些好了。

职业资源

推荐了重要的求职、职涯规划以及教育方面的资源。

蓝色理想经典论坛 - 企业招聘

提供了大量企业招聘信息，提供的职位主要集中在网站的设计师和开发人员。

地址：http://bbs.blueidea.com/forum-19-1.html

蓝色理想简介

成立于 1999 年 10 月。国内最大的设计类站点之一。重要的网站设计与开发人员社区。

除了人气极高的经典论坛，还提供经典作品集、经典桌面以及 Think.Pages 等多种产品。

ZDNet China 至顶网 - CIO 频道

CIO 的英文全称是Chief Information Officer。中文意思是首席信息官或信息主管

至顶网的 CIO 频道是 CIO 职业发展的重要工具。

地址：http://cio.zdnet.com.cn/

职涯发展栏目：http://cio.zdnet.com.cn/cio/more/3_842.shtml

ZDNet China 至顶网简介

国内最大的也是唯一企业级 IT 门户，著名科技专业品牌 ZDNet 旗下的网站。

网站平台内容已全面覆盖到 IT 从业者关注的相关内容，包含新闻、网络安全、存储、软件、产品，以及软件下载与技术白皮书下载服务。

重要的 IT 教育站点

CSDN : http://www.csdn.net/

中国 IT 实验室 : http://www.chinaitlab.com/

达内学院 : http://www.yc-edu.org/

Web 媒体

Web 多媒体 简介

多媒体（**Multimedia**）指图片、声音、音乐、动画和视频。

现代的 **web** 浏览器支持多种多媒体格式。

什么是多媒体？

多媒体是我们可以看到和听到的一切：文本、书籍、图片、音乐、声音、CD、视频、DVD、档案、电影等等。

多媒体以多种方式存在。在因特网上，您会发现很多被嵌入网页中的元素，并且今天的 web 浏览器已支持多种多媒体格式。

在本教程中，您会学到不同的多媒体格式，以及如何您的网页中使用它们。

浏览器支持

第一批因特网浏览器仅支持文本，甚至被限制为单一颜色的单一字体。

随后的 web 浏览器支持色彩、字体以及文本样式，并增加了对图像的支持。

不同的浏览器对声音、动画以及视频的处理方式是各不相同的。某些元素被内联处理，某些要求插件，而某些则要求 ActiveX 控件。

您将在下面的章节学到这方面的知识。

多媒体格式

多媒体元素（比如声音或视频）被存储在媒体文件中。

识别媒介类型的最通用的方法是查看文件的扩展名。

当浏览器获知文件的后缀是 .htm 或者 .html，它将假定文件是 HTML 页面。.xml 后缀表示 XML 文件，而 .CSS 后缀表示样式表。

图像格式是通过 .gif 或 .jpeg 后缀来识别的。

多媒体元素同样拥有带有不同后缀的文件格式。

在下面的章节，您将学到更多有关媒体文件后缀的知识。

多媒体 音频格式

声音可通过多种格式进行存储。

MIDI 格式

MIDI 是一种在电子音乐设备（比如合成器与 PC 声卡）之间传送音乐信息的格式。

MIDI 格式由音乐行业在 1982 年发明。MIDI 格式的伸缩性很强，可用于从极其简单的到非常专业的各种音乐制作。

MIDI 文件不包含被采样的声音，而是一系列可被 PC 的声卡解释的数字音乐指令（音符）。

MIDI 的劣势是它无法记录声音（仅能记录音符）。或者换句话说：它不能存储歌曲，仅能存储曲调。

点击这里演奏 The Beatles。

MIDI 格式的优势是，由于它只包含指令（音符），MIDI 文件可以非常之小。上面的例子仅有 23K，但是却可以演奏将近 5 分钟。

MIDI 格式得到了大范围的平台上许多不同的软件系统的支持。MIDI 文件也得到了所有最流行的因特网浏览器的支持。

以 MIDI 格式存储的音频的格式是 .mid 或 .midi。

RealAudio 格式

RealAudio 格式由 Real Media 针对因特网开发。此格式也支持视频。

此格式允许低带宽下的音频流（在线音乐、因特网广播）。由于低带宽的优先级，其品质往往被削弱。

以 RealAudio 格式存储的音频，其扩展名是 .rm 或 .ram。

AU 格式

AU 格式得到了大范围的平台上许多不同的软件系统的支持。

以 AU 格式存储的音频，其扩展名是 .au。

AIFF 格式

AIFF 格式 (Audio Interchange File Format) 是由 Apple 开发的。

AIFF 不是跨平台的，也不被所有的 web 浏览器支持。

以 AIFF 格式存储的音频，其扩展名是 aif 或 .aiff。

SND 格式

SND (Sound) 由 Apple 开发。

SND 不是跨平台的，也不被所有的 web 浏览器支持。

以 SND 格式存储的音频，其扩展名是 .snd。

WAVE 格式

WAVE (waveform) 格式由 IBM 和 Microsoft 开发。

它得到了所有运行 Windows 以及几乎所有最流行的 web 浏览器的支持。

以 WAVE 格式存储的音频，其扩展名是 .wav。

MP3 格式 (MPEG)

MP3 文件实际上是 MPEG 文件。但是起初，MPEG 格式是由运动图像专家组（Moving Pictures Experts Group）针对视频进行开发的。我们可以这么说，MP3 文件是 MPEG 视频格式的组成部分。

MP3 是音乐记录方面最流行的音频格式之一。MP3 编码系统结合了高压缩（小巧的文件）和高品质的优点。期望未来所有的系统都支持它。

以 MP3 格式存储的音频，其后缀是 .mp3, 或 .mpga（针对 MPG Audio）。

使用哪种格式？

在因特网上，WAVE 格式是最流行的格式之一，且得到了所有流行的浏览器的支持。如果您希望所录制的声音可用于所有的访问者，就应该使用 WAVE 格式。

MP3 格式则是一种新近来临的音频格式。如果您的网站与音乐有关，MP3 格式应该也是不错的选择。

多媒体 视频格式

视频可通过多种格式进行存储。

AVI 格式

AVI 格式 (Audio Video Interleave) 由微软开发。

AVI 格式得到了所有运行 Windows 的计算机的支持，以及绝大多数最流行的浏览器。在因特网上，它是非常普遍的一种格式，但是并不总能在非 Windows 的计算机上播放。

以 AVI 格式存储的视频，其扩展名是 .avi

Windows 媒介格式

Windows Media 格式由微软开发。

在因特网上，Windows Media 是非常普遍的一种格式，但是如果不安装一个额外的组件（免费），Windows Media 格式的影片无法在在非 Windows 的计算机上播放。由于没有可用的的播放器，某些后来的 Windows Media 影片在所有非 Windows 的计算机上都无法播放。

以 Windows Media 格式存储的视频，其扩展名是 .wmv。

MPEG 格式

MPEG (Moving Pictures Expert Group) 格式是因特网上最流行的格式。它是跨平台的，且得到了所有最流行的浏览器的支持。

以 MPEG 格式储存的视频，其扩展名是 .mpg 或 .mpeg。

QuickTime 格式

QuickTime 格式由 Apple 开发。

在因特网上 QuickTime 是一种普遍的格式，但是如果不安装额外的组件，QuickTime 格式的影片无法在 Windows 计算机上播放。

以 QuickTime 格式存储的视频，其后缀是 .mov。

RealVideo 格式

RealAudio 格式由 Real Media 针对因特网开发。

此格式运行低带宽下的视频流（在线视频、因特网电视）。由于其低带宽的优先级，视频质量往往会被削弱。

以 RealVideo 格式存储的视频，其后缀是 .rm 或 .ram。

Shockwave (Flash) 格式

Shockwave 格式由 Macromedia 开发。

Shockwave 格式需要额外的组件才能播放。该组件在最新版本的 Netscape 和 Internet Explorer 中是预安装的。

以 Shockwave 格式存储的视频，其扩展名是 .swf。

在 Web 上播放音频

根据您所使用的 **HTML** 元素，音频可“内联地”或通过某种“助手”进行播放。

内联音频

当音频被包含在网页中，或作为网页的一部份，它就被称为内联音频。

通过使用 <bgsound> 元素或 <img> 元素，可向网页添加内联音频。

如果你计划在 web 应用程序中使用内联音频，您需要清楚一点，就是许多人对内联音频非常讨厌。也请注意，一些用户也许已经在他们的浏览器中关闭了内联音频的选项。

我们的建议是，最近仅仅在用户希望听到声音的地方包含内联音频。比方说在用户打开页面后，点击某个链接来收听一段录音。

使用助手 （**Plug-In**，插件）

助手应用程序，是一种可通过浏览器启动来“帮助”浏览器播放音频的程序。助手应用程序也称为插件（Plug-Ins）。

助手应用程序可通过使用 <embed> 元素来启动，或者 <applet> 元素及 <object> 元素。

使用助手应用程序的一项巨大优势是，允许用户控制播放器的某些设置。

大多数助手应用程序允许手动地或通过编程控制音量设置以及播放功能，比如回放、暂停、停止和播放。

使用 **<bgsound>** 元素

Internet Explorer 支持 <bgsound> 元素。

该元素的作用是为网页提供背景音：

```
<bgsound src="beatles.mid" />
```

上面的代码片断为网页设置了一个 MIDI 文件作为背景音乐。

您可在本教程的最后一节找到 <bgsound> 元素的属性列表。

注释：<bgsound> 元素不是标准的 HTML 或 XHTML 元素。仅有 Internet Explorer 支持该元素。

使用 <img> 元素

Internet Explorer 支持 <img> 元素中的 dynsrc 属性。

该元素的作用是在网页中嵌入多媒体元素：

```
<img dynsrc="horse.wav" />
```

上面的代码片断为网页设置了一个嵌入的 WAVE 文件。

注释：dynsrc 属性不是标准的 HTML 或 XHTML 元素。仅有 Internet Explorer 支持该属性。

使用 <embed> 元素

Internet Explorer 和 Netscape 都支持 <embed> 元素。

该元素的作用是在网页中嵌入多媒体元素：

```
<embed src="beatles.mid" />
```

上面的代码片断为网页设置了一个嵌入的 MIDI 文件。

您可在本教程的最后一节找到 <embed> 元素的属性列表。

注释：Internet Explorer 和 Netscape 都支持 <embed> 元素，但它不是标准的 HTML 或 XHTML 元素。万维网联盟 (W3C) 推荐使用 <object> 元素来代替它。

使用 <object> 元素

Internet Explorer 和 Netscape 都支持 <object> 元素。

该元素的作用是在网页中嵌入多媒体元素：

```
<object
classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95">
<param name="FileName" value="liar.wav" />
</object>
```

上面的代码片断在网页设置了一个嵌入的的 WAVE 文件。

您可在本教程的最后一节找到 <object> 元素的属性列表。

使用超链接

如果网页包含了一个指向某个媒介文件的超链接，大多数浏览器都会使用“助手程序”来播放该文件：

```
<a href="beatles.mid">
点击此处来播放 the Beatles
</a>
```

上面的代码片段设置了一个指向 MIDI 文件的链接。如果用户点击该链接，浏览器将启动助手程序（比如 Windows Media Player）来播放该 MIDI 文件。

在 Web 上播放视频

根据您所使用的 **HTML** 元素，视频可“内联地”或通过某种“助手”进行播放。

内联视频（Inline Videos）

当视频被包含在网页中，或作为网页的一部份，它就被称为内联视频。

通过使用 <img> 元素，可向网页添加内联视频。

如果你计划在 web 应用程序中使用内联视频，您需要清楚一点，就是许多人对内联视频非常讨厌。也请注意，一些用户也许已经在他们的浏览器中关闭了内联视频的选项。

我们的建议是，最近仅仅在用户希望听到声音的地方包含内联视频。比方说在用户打开页面后，点击某个链接来观看视频。

使用助手（Plug-In，插件）

助手应用程序，是一种可通过浏览器启动来“帮助”浏览器播放视频的程序。助手应用程序也称为插件（Plug-Ins）。

助手应用程序可通过使用 <embed> 元素来启动，或者 <applet> 元素及 <object> 元素。

使用助手应用程序的一项巨大优势是，允许用户控制播放器的某些设置。

大多数助手应用程序允许手动地或通过编程控制音量设置以及播放功能，比如回放、暂停、停止和播放。

使用 <img> 元素

Internet Explorer 支持 <img> 元素中的 dynsrc 属性。

该元素的作用是在网页中嵌入多媒体元素：

```
<img dynsrc="video.avi" />
```

上面的代码片断为网页设置了一个嵌入的 AVI 文件。

注释：dynsrc 属性不是标准的 HTML 或 XHTML 元素。仅有 Internet Explorer 支持该属性。

使用 <embed> 元素

Internet Explorer 和 Netscape 都支持 <embed> 元素。

该元素的作用是在网页中嵌入多媒体元素：

```
<embed src="video.avi" />
```

上面的代码片断为网页设置了一个嵌入的 AVI 文件。

您可在本教程的最后一节找到 <embed> 元素的属性列表。

注释：Internet Explorer 和 Netscape 都支持 <embed> 元素，但它不是标准的 HTML 或 XHTML 元素。万维网联盟 (W3C) 推荐使用 <object> 元素来代替它。

使用 **<object>** 元素

Internet Explorer 和 Netscape 都支持 <object> 元素。

该元素的作用是在网页中嵌入多媒体元素：

```
<object data="video.avi" type="video/avi" />
```

上面的代码片断在网页设置了一个嵌入的的 AVI 文件。

您可在本教程的最后一节找到 <object> 元素的属性列表。

使用超链接

如果网页包含了一个指向某个媒介文件的超链接，大多数浏览器都会使用“助手程序”来播放该文件：

```
<a href="video.avi">点击此处来播放视频文件</a>
```

上面的代码片段设置了一个指向 AVI 文件的链接。如果用户点击该链接，浏览器将启动助手程序（比如 Windows Media Player）来播放该 AVI 文件。

Windows 多媒体格式

Windows 媒介文件拥有这些后缀：**.asf**、**.asx**、**.wma,** 以及 **.wmv**。

ASF 格式

ASF 格式 (Advanced Streaming Format) 是专门为在因特网上运行而设计的。

ASF 文件包含音频、视频、幻灯片展示以及同步事件。

ASF 文件能够被深度压缩，同时能够以连续的数据流进行传输（在线电视和广播）。该格式的文件可以是任何大小，同时能够被压缩以适应不同的带宽（连接速度）。

ASX 格式

ASX (Advanced Stream Redirector) 文件并非媒介文件，而是元数据文件。

元数据文件提供有关文件的信息。ASX 文件是用于描述多媒体内容的纯文本文件：

```
<ASX VERSION="3.0">
<Title>Holiday 2001</Title>
<Entry>
    <ref href="holiday-1.avi"/>
</Entry>
<Entry>
    <ref href="holiday-2.avi"/>
</Entry>
<Entry>
    <ref href="holiday-2.avi"/>
</Entry>
</ASX>
```

上面的文件描述了三个多媒体文件。当 ASX 文件被播放器读取时，播放器可播放被描述的文件。

WMA 格式

WMA (Windows Media Audio) 格式是由微软开发的音频格式。

WMA 的设计目标是处理各种类型的音频内容。该格式的文件能够被深度压缩，且能够传输连续的数据流（在线广播）。WMA 文件可以是任何大小，同时能够被压缩以适应不同的带宽（连接速度）。

WMA 格式与 ASF 格式类似。（请参阅上面的内容）。

WMV 格式

WMV (Windows Media Video) 格式是由微软开发的一种视频格式。

WMV 的设计目标是处理各种类型的视频内容。该格式的文件能够被深度压缩，且能够传输连续的数据流（在线广播）。WMA 文件可以是任何大小，同时能够被压缩以适应不同的带宽（连接速度）。

WMV 格式与 ASF 格式类似。（请参阅上面的内容）。

其他 Windows Media 格式

WAX (Windows Media Audio Redirector) 文件与 ASX 文件非常相似，不过旨在描述音频文件（.wma 文件）。

WMP (Windows Media Player) 文件和 WMX 是微软供未来使用的预留文件类型。

多媒体教程 - GIF 图像

GIF 是在 **Web** 上使用的主要图像格式之一。

本文详细讲解了 **GIF** 图像的特性和使用技巧。

理解图像格式

无论是 HTML 还是 XHTML 都没有规定图像的官方格式。然而，流行的浏览器却专门规定了一定的图像格式：通常情况下是 GIF 和 JPEG。其他多媒体格式大多需要特殊的辅助应用程序，每个浏览器的使用者都要去获得、安装并正确地操作这些应用程序，这样才能看到或听到这些特殊的文件。所以，GIF 和 JPEG 成为在 Web 上的实际标准也就不令人觉得奇怪了。

在 Web 出现以前，这两种图像格式已经得到了广泛使用，所以有大量支持软件可以帮助我们以这两种格式创建图像。然而，这两种格式各自有其优缺点，有些浏览器会利用其特性来实现特殊的显示效果。

GIF

GIF 格式指的是图像交换格式（Graphics Interchange Format，GIF），该格式最初是 CompuServe 为其在线服务用户传输图像而开发的。

GIF 格式的特性

GIF 格式有很多特性，因此在 HTML/XHTML 中十分普及。

首先，它的编码技术在许多平台上都可以使用。所以，通过适当地 GIF 解码软件（大多数浏览器都含有这种软件），在 Macintosh 上创建并组成 GIF 文件的图像，在基于 Windows 的 PC 上也可以毫不费力地加载、解码并查看。

GIF 格式的第二个特性是，它采用了一种特殊的压缩技术，可以显著减小图像文件的大小，从而得以在网络上更快地进行传输。而 GIF 压缩是“无损”压缩，也就是说，图像中原来的数据都不会发生改变或丢失，所以解压缩并解码后的图像与原来的图像完全一样。

此外，GIF 图像还非常容易实现动画效果。

GIF 格式的版本和颜色

尽管 GIF 图像文件都用 .gif （或者 .GIF）作为文件名后缀，实际上却有两个 GIF 版本：原始的 GIF87 和 GIF89a，后者支持很多新特性，包括透明背景、交叉存储和动画等，这些特性在 Web 创作者中的使用十分普及。

现在流行的浏览器都支持这两种 GIF 格式，它们都是通过同一种方案来把 8 位的像素值映射到一个颜色表当中，这样每个图像最多可以有 256 种颜色。

大多数 GIF 图像实际颜色的数目更少，有些特殊工具（比如 Macromedia 的 Fireworks）可以在更为精细的图像中简化这些颜色。通过简化颜色，可以创建更小的颜色映像并且强化像素冗余，来使文件压缩得更多，从而使下载速度更快。

然而，由于颜色数目有限，用 GIF 编码的图像并不是任何时候都适用，尤其是对那些具有照片一样逼真效果的图片来说。GIF 可以用来创建非常好看的图标和颜色不多的图像及图画。

即使是非常挑剔的创作者也会选择 **GIF**

因为大多数图形浏览器都明确地支持 GIF 格式，因此它现在是 Web 上接受面最广泛的图像编码格式。内联图像和外部链接图像都可以使用这种格式。如果你在选择图像格式方面犹豫不决，使用 GIF 肯定没错。它几乎在所有情况下都可以正常使用。

GIF 图像的技巧

GIF 图像有三种特殊的技巧：隔行扫描（interlacing）、透明性（transparency）和动画（animation）。

隔行扫描

通过隔行扫描，GIF 图像可以在屏幕上一下子显现出来，而不是从上到下逐步地显示。一般情况下，用 GIF 编码的图像是像素数据从图像的顶部到底部顺次、逐行排列的一个序列。所以，普通的 GIF 图像在屏幕上显示时，就好像一下子拉开窗帘，而具有隔行扫描的 GIF 在显示时，则像卷起百叶窗一样。这是因为像素数据的序列是每隔 4 行就交错一次。用户只需要用下载并显示一整幅图像的四分之一时间，就可以看到一个从上到下非常完整的图像，虽然它很模糊。这个只完成了四分之一的图像通常已经足够清楚了，这样那些网络连接速度较慢的用户就能够判断出，是否有必要花时间下载图像其余的部分。

尽管所有的图形浏览器都能够显示隔行扫描 GIF，但并不是所有浏览器都可以显示出隔行扫描那种逐渐清晰的效果。即使是那些可以实现这种效果的浏览器，用户还可以通过选择在图像完全下载并解码后再显示，来抑制这种效果。老式浏览器总是当图像完全下载并解码后才会显示，所以根本不支持这种效果。

透明性

GIF 图像（实际上是 GIF89a 格式的图像）另外一种常见的效果，是它可以让图像的一部分变成透明效果，这样图像下面的内容（通常是浏览器的窗口背景）就可以透过透明部分显示出来。透明的 GIF 图像在它的颜色映射里专门用一种颜色作为背景颜色，从而让显示窗口的背景透过来。通过仔细地剪切图像的大小和选择一种接近纯色的背景颜色，透明图像可以制作成看上去好像完全镶嵌在网页中，或者是浮动在上面的效果。

透明 GIF 图像适合于任何希望融入文档当中但又不希望看上去是个方块的图形。透明的 GIF 徽标十分常见，如透明图标或者印刷符号等 - 任何具有任意的自然形状的图像都可以使用这种效果。还可以在传统文本中插入透明的内联图像，以便在其中显示特殊的字符符号。

GIF 图像的透明效果有一个不好的地方，就是如果把它包含在超链接锚（<a>）标签里面而没有去掉它的边框，或者用框架专门将它括起来时，它看上去会十分糟糕。而且其他内容会围绕图像的矩形边框显示，而不是靠近图像的不透明边框。这样的结果就是不必要地把图像隔离出来，或者使网页看上去非常古怪。

动画

GIF89a 格式图像的第三个独特之处是，它可以实现简单的逐帧动画。通过使用特殊的 GIF 动画工具软件，就可以把一系列 GIF 图像放在一个单独的 GIF89a 文件中。浏览器会相继显示文件中的每个图像，就像我们小时候曾经玩过的（甚至画过的）那种通过快速翻页产生动画效果的小册子。在 GIF 文件中，每个图像之间都具有特殊的控制部分，可以用来设置浏览器从头到尾显示整个序列（循环）的次数，每两个图像之间停顿的时间，以及在浏览器显示后面一个图像之前是否从背景中抹去图像空间，等等。通过把这些特性与那些 GIF 通常具有的特性（包括单独的颜色表、透明性、隔行扫描等）结合起来，就可以创造出非常有吸引力而且非常精致的图像。

简单的 GIF 动画之所以具有强大的效果，还有另外一个重要的原因：不需要特地为 HTML 文档编写程序就可以获得动画效果。但它也有一个非常大的毛病，那就是它局限在一些象图标大小般很小的图形中，或者是只占据浏览器窗口中很窄一条的图形当中：即使你非常谨慎地没有在连续的动画单元中重复静态部分，GIF 动画也是非常容易变得很大。这样，如果文档中包含了多个动画，那么下载这些图像的拖延时间可能会令用户非常反感。如果说有什么特性值得我们非常小心仔细地对待而不至于滥用的话，那就是 GIF 动画。

总结

GIF 的所有技巧 - 隔行扫描、透明性和动画 - 都不是随随便便就可以获得的，它们都需要特殊的软件来准备这些 GIF 文件。现在很多图像软件都可以把用户创建的或者从外部获得的图像保存为 GIF 格式，而且大多数都支持透明效果，还可以实现 GIF 文件的隔行扫描效果。还有非常多专门为这些需要而设计的大量共享软件或免费软件，包括实现 GIF 动画的软件程序等。

多媒体教程 - JPEG 图像

JPEG 是在 **Web** 上使用的主要图像格式之一。

本文讲解 **JPEG** 图像的概念和特性。

理解图像格式

无论是 HTML 还是 XHTML 都没有规定图像的官方格式。然而，流行的浏览器却专门规定了一定的图像格式：通常情况下是 GIF 和 JPEG。其他多媒体格式大多需要特殊的辅助应用程序，每个浏览器的使用者都要去获得、安装并正确地操作这些应用程序，这样才能看到或听到这些特殊的文件。所以，GIF 和 JPEG 成为在 Web 上的实际标准也就不令人觉得奇怪了。

在 Web 出现以前，这两种图像格式已经得到了广泛使用，所以有大量支持软件可以帮助我们以这两种格式创建图像。然而，这两种格式各自有其优缺点，有些浏览器会利用其特性来实现特殊的显示效果。

JPEG

联合图像专家组（Joint PhotograPhic ExPerts Group，JPEG）是开发我们现在所使用的 JPEG 图像编码格式的标准化组织。

和 GIF 一样，JPEG 图像也是独立于平台的，而且为了通过数字通信技术来高速传播，而专门进行了压缩。和 GIF 不一样的是，JPEG 支持数以万计的颜色，可以显示更加精细而且像照片一样逼真的数字图像。

JPEG 使用的是特殊的压缩算法，从而可以实现非常高的压缩比。例如，把 200 KB 大小的 GIF 图像压缩到只有 30 KB 大小的 JPEG 图像，这种情况非常普通。为了达到这样惊人的压缩率，JPEG 要损失一些图像数据。然而，通过专门的 JPEG 工具可以调整这个“损失率”，这样，尽管压缩后的图像和原来的图像并不完全一样，但它们可以非常接近，以至于大多数人都无法分辨出之间的差别。

尽管 JPEG 对照片来说是一个不错的选择，但对插图（illustration）来说就不那么合适了。JPEG 使用的压缩和解压缩算法使得它在处理大范围的颜色块时，会留下很明显的人工痕迹。所以，如果你想显示出用线条描绘的图画，GIF 也许更适合一些。

JPEG 格式通常由 .jpg （或者 .JPG）文件名来结尾，现在，几乎所有图形浏览器都可以识别这种格式。只有在极少的情况下才可能遇到那些无法直接显示 JPEG 图像的老式浏览器。

多媒体教程 - 在 Web 上使用图像

本文详细讲解在什么条件下使用图像和文本、如何加快图像的下载以及选择合适的图像格式。

何时使用图像

对于大多数图片来说，一张图片可能胜过千言万语。但还是要注意，没有人会注意那些饶舌的人。首先，也是最重要的一点，是要把文档的图形作为可视化工具，而不是将其作为无缘无故的装饰。它们应当支持文本的内容，并帮助读者在文档中导航。使用图像可使文档内容更清楚，还可以为文档加注释或示例。支持内容的照片、图表、曲线图、地图和图画等都是很自然的、很合适的选择项。例如，产品的照片对于在线目录和购物指南来说是非常关键的组成部分。还有具有链接功能的图标和印刷符号，包括具有动画效果的图像等，都可以是导向内容或者外部资源的非常有效的可视向导。如果某个图像对文档没有起到任何上述作用，那就应该把它丢到一边去！

在考虑向文档添加图像时，另外一个重要的考虑因素，就是在通过网络，尤其是通过调制解调器连接传输这个文档时，检索方面所带来的时间延迟。一般的普通文档最多可以容纳 10-15 千字节，而一个图像可以轻易地达到数百千字节大小。而且一个文档的总下载时间不仅仅是它所有部分加起来的总和，还要考虑网络负载所带来的延迟。

根据连接的速度（也就是带宽，*bandwidth*，通常用 bps 或者 b/s 来表示）和可能减慢连接速度的网络阻塞情况，要下载一个包含 100 KB 图像的单独文档，可能要在凌晨一两点，当大多数人还在睡觉时，用一个 57.6 Kbps 的调制解调器连接花大约 15 秒钟左右的时间来完成下载，也有可能在中午的时候用一个 9600 bps 调制解调器花上超过 10 分钟的时间来下载。您得到这张图片了吗？

当然，图片和其他多媒体的使用，会促使因特网服务提供商不断追求更快、更好、更加健壮的方式来提供 Web 内容。不久，56 Kbps 调制解调器连接就会像马和马车一样退出历史舞台（就像 9600 bps调制解调器那样），它会被电缆调制解调器和 ADSL 这样的新技术所取代。实际上，大多数连接很快就会超过 1Mbps 的速率。

随着价格的降低，网络的使用会不断增加，于是就带来了阻塞的问题。如果你正在试图访问一个超负荷运转的服务器，那么不管你的网络连接速度有多快，都无法正常进行访问。

何时使用文本

文本并没有过时。对于某些用户来说，文本是他们文档中唯一可以访问的部分。我们建议，在大多数情况下，文档应当能够被任何人访问，包括那些无法浏览图像、或者那些为了改善网络连接性能而禁用浏览器自动下载图像功能的用户。虽然向文档中加入图像的需求可能会非常强烈，但是有些时候，纯文本文档确实会更有意义。

从其他格式转换为 Web 页面的文档很少含有嵌入式图像，而参考文献和其他一些严肃的内容，通常都是完全可用的纯文本形式。

在访问速度非常重要的时候，应该创建纯文本文档。如果你知道用户可能争着去获取你的文档，就应该在文档中避免使用图像，以适应这些用户的需要。在某些极端的情况下，您可能会提供一个主页（引导页），让用户有机会在您作品的两个副本之间做出选择：一个包含图像，另外一个则去掉了图像（流行浏览器都具有特殊的图像图标，来为那些有待下载的图像留出地方，而这些占位符可能会把文档的布局搞得一团糟，甚至变成一堆根本没有办法阅读的东西）。

如果希望你的文档可以很容易地被 Web 上众多的索引服务搜寻到，那么文本是最合适的形式 - 仅仅支持图像，而不支持装饰和不必要的图形。但是这些搜索引擎通常会忽略图像的存在。如果页面的主要内容是通过图像来提供的，那么在线 Web 目录中有关你的文档的信息就会很少。

加快图像的下载速度

除了谨慎地选择要包含在文档中的内容外，还有许多方法可以改善由图像带来的负载和延迟问题：

保持简单性

一个全屏的 24 位彩色图形，即使经过标准格式（例如 GIF 或者 JPEG 等）的数字压缩减少了尺寸，它还是会侵占大部分网络带宽。因此，最好使用各种图像管理工具来优化图像的尺寸，并将颜色的数目减为最少的像素数。简化你的绘图，也不要使用那些风景照片，并且避免在图像中出现大块的空白背景，和不必要的边框以及其他占用空间的元素。还要避免使用抖动效果（把两种相近的颜色混合起来已获得第三种颜色），这种技术会极大地降低图像的可压缩性。相反，要尽量使用大面积一致的颜色，因为用 GIF 或者 JPEG 格式可以很容易地对它们进行压缩。

重复利用图像

这一点对于图标和 GIF 动画尤其适用。大多数浏览器会在本地存储器把引入的文档成分进行缓存，这样在获得数据时可以更快，而且使用的网络连接也更少。对于较小的 GIF 动画文件，则要试着准备每个连续的图像，以便只更新那些在动画中发生了改变的部分，而不是刷新整个图像（这样也可以加速动画本身的显示）。

分割大文档

这是一个包含图像的一般原则。很多小的文档片段是用超链接（当然是用它了！）和有效的目录来组织到一起的，与整个的大块文档相比，这样可以让用户觉得更容易接受一些。一般来讲，人们宁愿在几个页面间翻来翻去，也不愿意浪费光阴等待下载一个大文档（这和电视频道浏览综合症有点相似）。一条好的经验是把每个文档保持在大小为 50 KB 左右，这样即使读者使用的是最慢的连接也不会感到厌烦。

必要时隔离大图形

为很大的图像专门提供一个链接，该链接可能是一个图像的缩略词，让读者决定是否需要花时间下载整幅图像，以及什么时候下载。而且，由于这样的图像不像内联图像那样和文档中的其他元素混在一起，因此很容易标识并保存在本地存储器上，以供日后研究使用。

指定图像的尺寸

最后，另外一种改善性能的方法是把图像矩形边界的高度和宽度都包含在它的标签里面。通过指定这些尺寸，就可以省去其他一些额外步骤，扩展功能的浏览器不必再用额外步骤下载、检验并计算图像在文档中的尺寸。然而，这种做法有一个不好的地方。如果用户关闭了自动下载图像的功能，浏览器还是会把为图像预留的空间以指定的尺寸显示出来。这样留给读者的通常是一个空的框架，虽然对于该问题还没有解决方案，但是我们还是鼓励您使用这些尺寸属性，因为我们鼓励一切能够改善网络性能的行为。

JPEG 还是 GIF？

如果图像的来源或者你的工具软件更倾向于某一种格式，您可能只能使用 JPEG 或者 GIF 图像中的一种。现在，这两种格式都得到了浏览器的广泛支持，所以不会存在用户能否浏览的问题。

然而，我们还是建议您使用一定的工具去创建或者转换这两种格式，以充分利用它们各自的功能。例如，可以把 GIF 的透明特性应用在图标和小的装饰符号上。而利用 JPEG 来压缩那些较大的颜色丰富的图像，以加快下载速度。

<img> 标签

<img> 标签允许在文档的当前文档流中引用或者插入图形图像。如需了解更多有关该标签的详细信息，请参阅：

- 教程：HTML 图像
- 参考手册：HTML <img> 标签

Object 元素



<object> 元素可支持多种不同的媒介类型，比如：

- 图片
- 音频
- 视频
- Other 对象

显示图片

你可以显示一幅图片：

```
<object height="100%" width="100%"
type="image/jpeg" data="audi.jpeg">
</object>
```

显示网页

您可以显示一张网页：

```
<object type="text/html" height="100%" width="100%"
data="http://www.w3school.com.cn">
</object>
```

播放音频

您可以播放音频：

```
<object
classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95">
<param name="FileName" value="liar.wav" />
</object>
```

播放视频

你可以播放视频：

```
<object
classid="clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95">
<param name="FileName" value="3d.wmv" />
</object>
```

显示日历

您可以显示日历：

```
<object width="100%" height="80%"
classid="clsid:8E27C92B-1264-101C-8A2F-040224009C02">
<param name="BackColor" value="14544622">
<param name="DayLength" value="1">
</object>
```

显示图形：

你可以显示图形：

```
<object width="200" height="200"
classid="CLSID:369303C2-D7AC-11D0-89D5-00A0C90833E6">
<param name="Line0001"
value="setFillColor(255, 0, 255)">
<param name="Line0002"
value="Oval(-100, -50, 200, 100, 30)">
</object>
```

显示 Flash

您还可以显示 flash 动画：

```
<object width="400" height="40"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com
/pub/shockwave/cabs/flash/swflash.cab#4,0,0,0">
<param name="SRC" value="bookmark.swf">
<embed src="bookmark.swf" width="400" height="40"></embed>
</object>
```

播放 QuickTime 影片

<object> 元素可播放 **QuickTime** 电影。

QuickTime 格式

QuickTime 格式由 Apple 开发。以 QuickTime 格式存储的视频的扩展名是 .mov。

在因特网上，QuickTime 是一种普遍的格式，但是如果没有额外的组件（免费），QuickTime 电影无法在非 Windows 的计算机上播放。

通过 object 元素，可轻松将播放 QuickTime 影片的代码添加到网页中。如果用户计算机中未安装 QuickTime 播放器，则可将 object 设置为自动安装 QuickTime 播放器。

解决方案

这些播放 QuickTime 影片的代码：

```
<object width="160" height="144"
classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
codebase="http://www.apple.com/qtactivex/qtplugin.cab">
<param name="src" value="sample.mov">
<param name="autoplay" value="true">
<param name="controller" value="false">

<embed src="sample.mov" width="160" height="144"
autoplay="true" controller="false"
pluginspage="http://www.apple.com/quicktime/download/">
</embed>

</object>
```

<object> 元素

object 元素的 width 和 height 属性应当匹配影片的尺寸（以像素计）。

classid 可唯一地标识要使用的播放器软件。它必须设置为 "clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"。该唯一编码标识了在电影播放之前必须安装在用户 PC 上的 ActiveX 控件。如果用户未安装该 ActiveX 控件，则浏览器将自动下载并安装它。

codebase 属性规定了基准路径，该路径用于解析由 classid、data 和 archive 属性规定的相对 URL。如果未规定，则其默认值是当前文档的基准 URL。注释：Internet Explorer 使用该属性来规定播放器的下载位置。该属性必须被设置为 "http://www.apple.com/qtactivex/qtplugin.cab"。此位置包含 QuickTime 播放器的最新版本。

src 参数指向电影文件。

如果您希望电影自动播放的话，请将 autoplay 参数设置为 "true"。

如果您不希望显示控制按钮，请将 controller 参数设置为 "false"。

<embed> 元素

可以添加 embed 元素，来支持那些不支持 object 元素的浏览器。可理解 object 元素的浏览器将忽略 embed 元素。那些支持 ActiveX 控件的新浏览器（Internet Explorer 5、6、7）将使用 object 元素，而其他较老的浏览器 (Netscape 4 and 5) 将使用 embed 元素。

embed 元素的 width 和 height 属性应当匹配影片的尺寸（以像素计）。

embed 元素的 autoplay 和 controller 属性应设置为与 object 元素的相关属性相同的值。

pluginspage 属性定义了播放器的下载路径，必须被设置为 "http://www.apple.com/quicktime/download/"。

播放 Real Video 影片

<object> 元素能够播放 **Real Video** 影片。

Real Video 格式

RealVideo 格式由 Real Media 开发。以 Real Video 格式存储的视频，其扩展名是 .rm 或 .ram。

该格式允许低带宽下的视频流（在线视频、因特网电视）。由于其低带宽优先权，往往会削弱视频质量。

解决方案

这是播放 Real Video 影片所需的代码：

```
<object width="320" height="240"
classid="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA">
<param name="controls" value="ImageWindow" />
<param name="autostart" value="true" />
<param name="src" value="male.ram" />
</object>
```

<object> 元素

object 元素的 width 和 height 属性应当匹配影片的尺寸（以像素计）。

classid 可唯一地标识要使用的播放器软件。它必须设置为 "clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"。该唯一编码标识了在电影播放之前必须安装在用户 PC 上的 ActiveX 控件。如果用户未安装该 ActiveX 控件，则浏览器将自动下载并安装它。

param 元素为播放器提供附加信息。

src 参数指向电影文件。

如果您希望电影自动播放的话，请将 autostart 参数设置为 "true"。

如果您不希望显示控制按钮，请将 controls 参数设置为 "ImageWindow"，如果希望显示所有控制按钮，请将该参数设置为 "All"。

Object 参考

属性	定义
classid	对象的唯一 id。
height	对象的高度。以像素或百分比计。
width	对象的宽度。以像素或百分比计。

Parameter 参考

属性	定义
src	RealAudio 或 RealVideo 片断的源。
controls	控件的可见性。（请参阅下文）。
console	链接多个控件的控制台名称。
autostart	自动播放。 (true | false).
nolabels	禁止控件窗口的 label 文本。
reset	重置播放列表控件 (true | false).
autogotoURL	如何处理 URL。(true | false)
True	指向浏览器的向前 URL 事件。
False	使用 VBScript 代之。

控件的值

值	显示
All	显示带有所有控件的完整播放器。
InfoVolumePanel	标题、作者、版权以及音量滑块。
InfoPanel	标题、作者以及版权。
ControlPanel	位置滑块、播放、暂停以及停止按钮。
StatusPanel	消息、当前时间位置以及片断长度。
PlayButton	播放和暂停按钮。
StopButton	停止按钮。
VolumeSlider	音量滑块。
PositionField	位置和片断长度。
StatusField	消息。
ImageWindow	视频图像。
StatusBar	状态、位置和频道。

Web 多媒体元素参考手册

<bgsound> 元素

属性	功能
id	A unique id for the element.
src	The location (URL) of the source file.
balance	The balance. (-10000=left, +10000=right).
loop	The number of loops. (-1=infinite).
volume	The volume. (0=max, -10000=min).

<embed> 元素

属性	定义
autostart	Automatic start. (true | false).
height	The height of the element in pixels or %.
hidden	The visibility of the element. (true | false).
src	The location (URL) of the source file.
width	The width of the element in pixels or %.

请同时参阅页面底部的样式化属性、通用 HTML 属性以及事件属性：

<applet> 元素

属性	定义
alt	An alternate text.
archive	The locations (URLs) of archive files.
code	The location (URL) of the applet code.
codebase	The base location (default URL) for all files.
height	The height of the applet in pixels or %.
name	The name of the applet.
object	A saved representation of the applet. Do not use.
width	The width of the applet in pixels or %.

请同时参阅页面底部的样式化属性、通用 HTML 属性以及事件属性：

<object> 元素

属性	定义
archive	The locations (URLs) of archive files.
classid	The location (URL) of the object.
codebase	The base path used to resolve relative URIs specified by the classid, data, and archive attributes.
codetype	The content type of the code.
data	The location (URL) of object data.
declare	Do not instantiate (execute) the object.
height	The height of the object in pixels or %.
name	The object's name.
standby	Text to display while object is loading.
tabindex	The position in the tab order
type	The content type of the object.
usemap	The location (URL) of an image map.
width	The width of the player in pixels or %.

请同时参阅页面底部的样式化属性、通用 HTML 属性以及事件属性：

<param> 元素

param 元素为 object 或 applet 元素定义参数。

属性	定义
id	A unique id for the element.
name	Parameter name.
type	Parameter content type.
value	Parameter value.
valuetype	Parameter value type.

样式化属性

注释：这些参数不被赞成使用。请使用样式取而代之。

属性	定义
align	The alignment of the object.
border	The border with in pixels.
hspace	The horizontal white-space (margin) in pixels.
vspace	The vertical white-space (margin) in pixels.

通用 HTML 属性

属性	定义
class	The element's class.
dir	The directionality of the element.
id	A unique id for the element.
lang	The language used by the element.
style	The element's style.
title	The elements title.

标准事件

事件	句柄
onclick	mouse clicked
ondblclick	mouse double clicked
onmousedown	mouse button pressed down
onmouseup	mouse button released
onmouseover	cursor moved onto the element
onmousemove	cursor moved within the element
onmouseout	cursor moved away from the element
onkeypressed	key pressed and released over the element
onkeydown	key pressed down over the element
onkeyup	key released over the element

Windows Media Player 参考手册

Windows Media Player

Windows Media Player 存在多个不同的版本。

在此，我们提供了不同版本的 class ID，以及参数列表。

为什么 class ID 会改变？

Windows Media Player 7 及更高版本的 class ID 是：*clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6.*

因特网上的许多地方把 class ID 声明为：*clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95.* 此 class ID 是一个老的版本，但是依然可以工作，这是因为其向后兼容性。但是如果您使用了老的 class ID，就无法使用增加到组件中的新特性了。

Windows Media Player 10

clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6 （与 WMP7 相同）

Windows Media Player 9

clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6 （与 WMP7 相同）

Windows Media Player 7

clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6

Windows Media Player 6.4

clsid:22D6F312-B0F6-11D0-94AB-0080C74C7E95

参数	默认值	描述
AudioStream	true	
AutoSize	true	

AutoStart	true	Sets if the player should start automatically
AnimationAtStart	true	Sets if an animation should show while the file loads
AllowScan	true	
AllowChangeDisplaySize	true	
AutoRewind	false	
Balance	false	
BaseURL		
BufferingTime	5	
CaptioningID		
ClickToPlay	true	Sets if the player should start when the user clicks in the play area
CursorType	false	
CurrentPosition	true	
CurrentMarker	false	
DefaultFrame		
DisplayBackColor	false	
DisplayForeColor	16777215	
DisplayMode	false	
DisplaySize	false	
Enabled	true	
EnableContextMenu	true	
EnablePositionControls	true	
EnableFullScreenControls	false	
EnableTracker	true	
Filename	*URL*	The URL of the file to play
InvokeURLs	true	
Language	true	
Mute	false	
PlayCount	1	
PreviewMode	false	
Rate	1	

SAMILang		
SAMIStyle		
SAMIFileName		
SelectionStart	true	
SelectionEnd	true	
SendOpenStateChangeEvents	true	
SendWarningEvents	true	
SendErrorEvents	true	
SendKeyboardEvents	false	
SendMouseClickEvents	false	
SendMouseMoveEvents	false	
SendPlayStateChangeEvents	true	
ShowCaptioning	false	
ShowControls	true	Sets if the player controls should show
ShowAudioControls	true	Sets if the audio controls should show
ShowDisplay	false	Sets if the display should show
ShowGotoBar	false	Sets if the GotoBar should show
ShowPositionControls	true	
ShowStatusBar	false	
ShowTracker	true	
TransparantAtStart	false	
VideoBorderWidth	false	
VideoBorderColor	false	
VideoBorder3D	false	
Volume	-200	
WindowlessVideo	false	

Windows Media Player 6 （更老的版本）

clsid:05589FA1-C356-11CE-BF01-00AA0055595A

参数	默认值	描述
Appearance	false	
AutoStart	false	
AllowChangeDisplayMode	true	
AllowHideDisplay	false	
AllowHideControls	true	
AutoRewind	true	
Balance	false	
CurrentPosition	false	
DisplayBackColor	false	
DisplayForeColor	16777215	
DisplayMode	false	
Enabled	true	
EnableContextMenu	true	
EnablePositionControls	true	
EnableSelectionControls	false	
EnableTracker	true	
Filename		
FullScreenMode	false	
MovieWindowSize	false	
PlayCount	1	
Rate	1	
SelectionStart	true	
SelectionEnd	true	
ShowControls	true	
ShowDisplay	true	
ShowPositionControls	false	
ShowTracker	true	
Volume	-200	

MIME 参考手册

MIME 类型

MIME (_M_ultipurpose _I_nternet _M_ail _E_xtensions) 是描述消息内容类型的因特网标准。

MIME 消息能包含文本、图像、音频、视频以及其他应用程序专用的数据。

官方的 MIME 信息是由 Internet Engineering Task Force (IETF) 在下面的文档中提供的：

- RFC-822 Standard for ARPA Internet text messages
- RFC-2045 MIME Part 1: Format of Internet Message Bodies
- RFC-2046 MIME Part 2: Media Types
- RFC-2047 MIME Part 3: Header Extensions for Non-ASCII Text
- RFC-2048 MIME Part 4: Registration Procedures
- RFC-2049 MIME Part 5: Conformance Criteria and Examples

不同的应用程序支持不同的 MIME 类型。

下面的参考手册是由 Microsoft Internet Information Server version 5 所支持的 MIME 类型列表。

按照内容类型排列的 Mime 类型列表

类型/子类型	扩展名
application/envoy	evy
application/fractals	fif
application/futuresplash	spl
application/hta	hta
application/internet-property-stream	acx
application/mac-binhex40	hqx
application/msword	doc
application/msword	dot
application/octet-stream	*
application/octet-stream	bin
application/octet-stream	class
application/octet-stream	dms

application/octet-stream	exe
application/octet-stream	lha
application/octet-stream	lzh
application/oda	oda
application/olescript	axs
application/pdf	pdf
application/pics-rules	prf
application/pkcs10	p10
application/pkix-crl	crl
application/postscript	ai
application/postscript	eps
application/postscript	ps
application/rtf	rtf
application/set-payment-initiation	setpay
application/set-registration-initiation	setreg
application/vnd.ms-excel	xla
application/vnd.ms-excel	xlc
application/vnd.ms-excel	xlm
application/vnd.ms-excel	xls
application/vnd.ms-excel	xlt
application/vnd.ms-excel	xlw
application/vnd.ms-outlook	msg
application/vnd.ms-pkicertstore	sst
application/vnd.ms-pkiseccat	cat
application/vnd.ms-pkistl	stl
application/vnd.ms-powerpoint	pot
application/vnd.ms-powerpoint	pps
application/vnd.ms-powerpoint	ppt
application/vnd.ms-project	mpp
application/vnd.ms-works	wcm
application/vnd.ms-works	wdb

application/vnd.ms-works	wks
application/vnd.ms-works	wps
application/winhlp	hlp
application/x-bcpio	bcpio
application/x-cdf	cdf
application/x-compress	z
application/x-compressed	tgz
application/x-cpio	cpio
application/x-csh	csh
application/x-director	dcr
application/x-director	dir
application/x-director	dxr
application/x-dvi	dvi
application/x-gtar	gtar
application/x-gzip	gz
application/x-hdf	hdf
application/x-internet-signup	ins
application/x-internet-signup	isp
application/x-iphone	iii
application/x-javascript	js
application/x-latex	latex
application/x-msaccess	mdb
application/x-mscardfile	crd
application/x-msclip	clp
application/x-msdownload	dll
application/x-msmediaview	m13
application/x-msmediaview	m14
application/x-msmediaview	mvb
application/x-msmetafile	wmf
application/x-msmoney	mny
application/x-mspublisher	pub
application/x-msschedule	scd

application/x-msterminal	trm
application/x-mswrite	wri
application/x-netcdf	cdf
application/x-netcdf	nc
application/x-perfmon	pma
application/x-perfmon	pmc
application/x-perfmon	pml
application/x-perfmon	pmr
application/x-perfmon	pmw
application/x-pkcs12	p12
application/x-pkcs12	pfx
application/x-pkcs7-certificates	p7b
application/x-pkcs7-certificates	spc
application/x-pkcs7-certreqresp	p7r
application/x-pkcs7-mime	p7c
application/x-pkcs7-mime	p7m
application/x-pkcs7-signature	p7s
application/x-sh	sh
application/x-shar	shar
application/x-shockwave-flash	swf
application/x-stuffit	sit
application/x-sv4cpio	sv4cpio
application/x-sv4crc	sv4crc
application/x-tar	tar
application/x-tcl	tcl
application/x-tex	tex
application/x-texinfo	texi
application/x-texinfo	texinfo
application/x-troff	roff
application/x-troff	t
application/x-troff	tr

application/x-troff-man	man
application/x-troff-me	me
application/x-troff-ms	ms
application/x-ustar	ustar
application/x-wais-source	src
application/x-x509-ca-cert	cer
application/x-x509-ca-cert	crt
application/x-x509-ca-cert	der
application/ynd.ms-pkipko	pko
application/zip	zip
audio/basic	au
audio/basic	snd
audio/mid	mid
audio/mid	rmi
audio/mpeg	mp3
audio/x-aiff	aif
audio/x-aiff	aifc
audio/x-aiff	aiff
audio/x-mpegurl	m3u
audio/x-pn-realaudio	ra
audio/x-pn-realaudio	ram
audio/x-wav	wav
image/bmp	bmp
image/cis-cod	cod
image/gif	gif
image/ief	ief
image/jpeg	jpe
image/jpeg	jpeg
image/jpeg	jpg
image/pipeg	jfif
image/svg+xml	svg
image/tiff	tif

image/tiff	tiff
image/x-cmu-raster	ras
image/x-cmx	cmx
image/x-icon	ico
image/x-portable-anymap	pnm
image/x-portable-bitmap	pbm
image/x-portable-graymap	pgm
image/x-portable-pixmap	ppm
image/x-rgb	rgb
image/x-xbitmap	xbm
image/x-xpixmap	xpm
image/x-xwindowdump	xwd
message/rfc822	mht
message/rfc822	mhtml
message/rfc822	nws
text/css	css
text/h323	323
text/html	htm
text/html	html
text/html	stm
text/iuls	uls
text/plain	bas
text/plain	c
text/plain	h
text/plain	txt
text/richtext	rtx
text/scriptlet	sct
text/tab-separated-values	tsv
text/webviewhtml	htt
text/x-component	htc
text/x-setext	etx

text/x-vcard	vcf
video/mpeg	mp2
video/mpeg	mpa
video/mpeg	mpe
video/mpeg	mpeg
video/mpeg	mpg
video/mpeg	mpv2
video/quicktime	mov
video/quicktime	qt
video/x-la-asf	lsf
video/x-la-asf	lsx
video/x-ms-asf	asf
video/x-ms-asf	asr
video/x-ms-asf	asx
video/x-msvideo	avi
video/x-sgi-movie	movie
x-world/x-vrml	flr
x-world/x-vrml	vrml
x-world/x-vrml	wrl
x-world/x-vrml	wrz
x-world/x-vrml	xaf
x-world/x-vrml	xof

按照文件扩展名排列的 **Mime** 类型列表

扩展名	类型/子类型
	application/octet-stream
323	text/h323
acx	application/internet-property-stream
ai	application/postscript
aif	audio/x-aiff
aifc	audio/x-aiff

aiff	audio/x-aiff
asf	video/x-ms-asf
asr	video/x-ms-asf
asx	video/x-ms-asf
au	audio/basic
avi	video/x-msvideo
axs	application/olescript
bas	text/plain
bcpio	application/x-bcpio
bin	application/octet-stream
bmp	image/bmp
c	text/plain
cat	application/vnd.ms-pkiseccat
cdf	application/x-cdf
cer	application/x-x509-ca-cert
class	application/octet-stream
clp	application/x-msclip
cmx	image/x-cmx
cod	image/cis-cod
cpio	application/x-cpio
crd	application/x-mscardfile
crl	application/pkix-crl
crt	application/x-x509-ca-cert
csh	application/x-csh
css	text/css
dcr	application/x-director
der	application/x-x509-ca-cert
dir	application/x-director
dll	application/x-msdownload
dms	application/octet-stream
doc	application/msword
dot	application/msword

dvi	application/x-dvi
dxr	application/x-director
eps	application/postscript
etx	text/x-setext
evy	application/envoy
exe	application/octet-stream
fif	application/fractals
flr	x-world/x-vrml
gif	image/gif
gtar	application/x-gtar
gz	application/x-gzip
h	text/plain
hdf	application/x-hdf
hlp	application/winhlp
hqx	application/mac-binhex40
hta	application/hta
htc	text/x-component
htm	text/html
html	text/html
htt	text/webviewhtml
ico	image/x-icon
ief	image/ief
iii	application/x-iphone
ins	application/x-internet-signup
isp	application/x-internet-signup
jfif	image/pipeg
jpe	image/jpeg
jpeg	image/jpeg
jpg	image/jpeg
js	application/x-javascript
latex	application/x-latex

lha	application/octet-stream
lsf	video/x-la-asf
lsx	video/x-la-asf
lzh	application/octet-stream
m13	application/x-msmediaview
m14	application/x-msmediaview
m3u	audio/x-mpegurl
man	application/x-troff-man
mdb	application/x-msaccess
me	application/x-troff-me
mht	message/rfc822
mhtml	message/rfc822
mid	audio/mid
mny	application/x-msmoney
mov	video/quicktime
movie	video/x-sgi-movie
mp2	video/mpeg
mp3	audio/mpeg
mpa	video/mpeg
mpe	video/mpeg
mpeg	video/mpeg
mpg	video/mpeg
mpp	application/vnd.ms-project
mpv2	video/mpeg
ms	application/x-troff-ms
mvb	application/x-msmediaview
nws	message/rfc822
oda	application/oda
p10	application/pkcs10
p12	application/x-pkcs12
p7b	application/x-pkcs7-certificates
p7c	application/x-pkcs7-mime

p7m	application/x-pkcs7-mime
p7r	application/x-pkcs7-certreqresp
p7s	application/x-pkcs7-signature
pbm	image/x-portable-bitmap
pdf	application/pdf
pfx	application/x-pkcs12
pgm	image/x-portable-graymap
pko	application/ynd.ms-pkipko
pma	application/x-perfmon
pmc	application/x-perfmon
pml	application/x-perfmon
pmr	application/x-perfmon
pmw	application/x-perfmon
pnm	image/x-portable-anymap
pot,	application/vnd.ms-powerpoint
ppm	image/x-portable-pixmap
pps	application/vnd.ms-powerpoint
ppt	application/vnd.ms-powerpoint
prf	application/pics-rules
ps	application/postscript
pub	application/x-mspublisher
qt	video/quicktime
ra	audio/x-pn-realaudio
ram	audio/x-pn-realaudio
ras	image/x-cmu-raster
rgb	image/x-rgb
rmi	audio/mid
roff	application/x-troff
rtf	application/rtf
rtx	text/richtext
scd	application/x-msschedule

sct	text/scriptlet
setpay	application/set-payment-initiation
setreg	application/set-registration-initiation
sh	application/x-sh
shar	application/x-shar
sit	application/x-stuffit
snd	audio/basic
spc	application/x-pkcs7-certificates
spl	application/futuresplash
src	application/x-wais-source
sst	application/vnd.ms-pkicertstore
stl	application/vnd.ms-pkistl
stm	text/html
svg	image/svg+xml
sv4cpio	application/x-sv4cpio
sv4crc	application/x-sv4crc
swf	application/x-shockwave-flash
t	application/x-troff
tar	application/x-tar
tcl	application/x-tcl
tex	application/x-tex
texi	application/x-texinfo
texinfo	application/x-texinfo
tgz	application/x-compressed
tif	image/tiff
tiff	image/tiff
tr	application/x-troff
trm	application/x-msterminal
tsv	text/tab-separated-values
txt	text/plain
uls	text/iuls
ustar	application/x-ustar

vcf	text/x-vcard
vrml	x-world/x-vrml
wav	audio/x-wav
wcm	application/vnd.ms-works
wdb	application/vnd.ms-works
wks	application/vnd.ms-works
wmf	application/x-msmetafile
wps	application/vnd.ms-works
wri	application/x-mswrite
wrl	x-world/x-vrml
wrz	x-world/x-vrml
xaf	x-world/x-vrml
xbm	image/x-xbitmap
xla	application/vnd.ms-excel
xlc	application/vnd.ms-excel
xlm	application/vnd.ms-excel
xls	application/vnd.ms-excel
xlt	application/vnd.ms-excel
xlw	application/vnd.ms-excel
xof	x-world/x-vrml
xpm	image/x-xpixmap
xwd	image/x-xwindowdump
z	application/x-compress
zip	application/zip